Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/US2004/029715

International filing date: 09 September 2004 (09.09.2004)

Document type: Certified copy of priority document

Document details: Country/Office: US
Number: 60/501,970
Filing date: 10 September 2003 (10.09.2003)

Date of receipt at the International Bureau: 14 March 2007 (14.03.2007)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

1583395

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

March 05, 2007

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE.

APPLICATION NUMBER: *60/501,970*
FILING DATE: *September 10, 2003*
RELATED PCT APPLICATION NUMBER: *PCT/US04/29715*

THE COUNTRY CODE AND NUMBER OF YOUR PRIORITY APPLICATION, TO BE USED FOR FILING ABROAD UNDER THE PARIS CONVENTION, IS *US60/501,970*

UNITED STATES PATENT AND TRADEMARK OFFICE · DEPARTMENT OF COMMERCE

Certified by

Jon W. Dudas

Under Secretary of Commerce
for Intellectual Property
and Director of the United States
Patent and Trademark Office

PTO/SB/16 (08-03)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

Express Mail Label No. EE353630837US

INVENTOR(S)		
Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
Richard	Farrell	Tyngsboro, MA

Additional inventors are being named on the 1 *separately numbered sheets attached hereto*

TITLE OF THE INVENTION (500 characters max)

TRANSPORT PROTOCOL OPTIMIZER

Direct all correspondence to: **CORRESPONDENCE ADDRESS**

☑ Customer Number: 28020

OR

☐ Firm or Individual Name	Malcolm D. Reid				
Address	Gray, Plant, Mooty, Mooty & Bennett, PA				
Address	P.O. Box 2906				
City	Minneapolis	State	MN	Zip	55402-0906
Country	USA	Telephone	612-343-5378	Fax	612-333-0066

ENCLOSED APPLICATION PARTS *(check all that apply)*

☑ Specification *Number of Pages* 67 ☐ CD(s), Number

☑ Drawing(s) *Number of Sheets* 37 ☐ Other (specify)

☐ Application Date Sheet. See 37 CFR 1.76

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT

☑ Applicant claims small entity status. See 37 CFR 1.27.

☐ A check or money order is enclosed to cover the filing fees.

☑ The Director is herby authorized to charge filing fees or credit any overpayment to Deposit Account Number: 50-0937

☐ Payment by credit card. Form PTO-2038 is attached.

FILING FEE Amount ($): 80.00

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☑ No.

☐ Yes, the name of the U.S. Government agency and the Government contract number are:

Respectfully submitted,

SIGNATURE 

Date 09/10/2003

REGISTRATION NO. 27,065 *(if appropriate)*

Docket Number: 93088

TYPED or PRINTED NAME Malcolm D. Reid

TELEPHONE 612-343-5378

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

This collection of information is required by 37 CFR 1.51. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 8 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Mail Stop Provisional Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PROVISIONAL APPLICATION COVER SHEET
Additional Pag

PTO/SB/16 (08-03)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Docket Number 93088

INVENTOR(S)/APPLICANT(S)		
Given Name (first and middle [if any])	Family or Surname	Residence (City and either State or Foreign Country)
David J. James	Reiland Signorelli	Maple Grove, MN Champlin, MN

Number 1 of 1

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

GRAY
PLANT
MOOTY

3400 City Center
33 South Sixth Street
Minneapolis, MN 55402-3796

612 343-2800
Fax: 612 333-0066
www.gpmlaw.com

Including The Law Firm Hall & Byers, P.A.
1010 West St. Germain Street, Suite 600
St. Cloud, MN 56301

320 252-4414
Fax: 320 252-4482
www.gpmlaw.com

Reply to Minneapolis
Malcolm D. Reid
612 343-5378
malcolm.reid@gpmlaw.com

September 10, 2003

VIA EXPRESS MAIL (LABEL # EE353630837US)

Mail Stop PROVISIONAL
Commissioner for Patents
P.O. Box 1450
Arlington, VA 22313-1450

Inventor: Farrell, Richard *et. al.*
Application No.: unknown
Date Filed: September 10, 2003
Title: TRANSPORT PROTOCOL OPTIMIZER
Attorney Docket No.: 93088

TRANSMITTAL LETTER

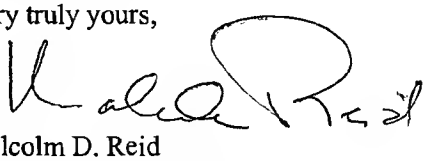
Dear Sir:

Enclosed for filing with the United States Patent and Trademark Office, please find:

1. Transmittal Letter, including mailing by Express Mail under 37 C.F.R. 1.10;
2. Provisional Application for Patent Transmittal Sheet (Form PTO/SB/16 – 2 Sheets);
3. Provisional Application papers (67 Sheets Specification, 37 Sheets of Figures); and,
4. Postage Paid Postcard Itemizing Above Documents.

Please charge Deposit Account 500937 for the required filing fees ($80.00) and any other additional fees required at this time. If you have any questions, please contact me.

Very truly yours,



Malcolm D. Reid
Registration No. 27,065
Attorney for Applicants

Certificate of Mailing by Express Mail under 37 C.F.R. 1.10

Express Mail label number: **EE353630837US**. Date of Deposit: September 10, 2003. I hereby certify that the following identified correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. 1.10 on the date indicated above and is addressed to Box Patent Application, Assistant Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450: Transmittal Letter, including mailing by Express Mail under 37 C.F.R. 1.10; Provisional Application for Patent Transmittal Sheet (Form PTO/SB/16 – 2 Sheets); Provisional Application papers (67 Sheets Specification, 37 Sheets of Figures); and, Postage Paid Postcard Itemizing Above Documents.

Judy Sigal
Judy Sigal

GP:1500326 v1

Title
Transport Protocol Optimizer

Description

Field of the Invention

The present invention generally relates to data communication protocols, and a means for optimizing the end-to-end throughput performance over TCP/IP networks.

Background of the Invention

There is an ever increasing demand for using Internet Protocol (IP) networks for more applications, particularly in the storage arena. Companies are faced with new, and more aggressive, requirements for providing Business Continuity and Disaster Recovery plans. To address this need, storage vendors are also introducing new products that can take advantage of existing network connectivity options to support the Business Continuity requirements. Coupled with this, new technology advances are resulting in faster communication paths being available.

However, the most prevalent communication protocol in use today, TCP/IP, was developed for Internet Protocol (IP) networks, but was not designed to efficiently handle today's networking speeds and workload requirements. Certain advances have been made in TCP/IP technology, but it is not yet universally deployed. Furthermore, deploying it requires additional efforts to be made, such as system tuning and changing applications. The present invention, Transport Protocol Optimizer, provides the capability of enhancing the performance of existing IP applications without requiring any application changes to be made.

Summary of the Invention

To meet this need, the present invention provides a method of optimizing IP traffic over a communications link. The method comprises:

- inserting a Transport Protocol Optimizer (TPO) on each of two Local Area Network (LAN) segments that are interconnected by a communications link.
- establishing a User Datagram Protocol (UDP) communication link between the two Transport Protocol Optimizers.
- directing IP packets to the Transport Protocol Optimizer on each of the LAN segments through standard IP routing mechanisms. The IP packets are generated as a result of communications applications executing on servers that are connected to these network segments.
- accepting the IP packets directed to the TPO based on a set of address and protocol filtering rules.
- rerouting the accepted packets to the IP address of the local TPO, while maintaining knowledge of the following fields from the rerouted packet: source IP address, source IP port, destination IP address, destination IP port, and protocol type.
- providing a local endpoint for the application TCP/IP connections for which packets are being rerouted to the TPO, by generating the required TCP/IP protocol to the application executing on the server that is connected to the same network segment as the source TPO.

- aggregating the accepted packet data into large buffers for subsequent transmission over the TPO communication link.
- sending the aggregated data over the TPO communication link, and providing a reliable data delivery mechanism.
- separating the aggregated packet data received by the destination TPO and restoring the following fields to the rerouted packets: source IP address, source IP port, destination IP address, destination IP port, and protocol type.
- delivering the packets to the original destination IP address on the destination network segment.
- providing a local endpoint for the application TCP/IP connections for which packets are being rerouted to the TPO, by generating the required TCP/IP protocol to the application executing on the server that is connected to the same network segment as the destination TPO.

A data communications system of one or more sets of peer-to-peer TCP/IP applications, a method of intercepting Internet Protocol (IP) packets generated by end-user networking applications, comprising the steps of:

- incorporating an IP Netfilter mechanism to gain visibility to IP packets
- identifying the IP packet headers to be intercepted, based on source IP address, source port number, destination IP address, destination port number, and protocol type
- redirecting the desired IP packets to the Transport Protocol Optimizer

A data communications system of one or more sets of peer-to-peer TCP/IP applications, a method of aggregating data from intercepted Internet Protocol (IP) packets that are generated by one or more end-user networking applications, and forming a Packet Driver message, comprising the steps of:

- an Internet Protocol (IP) header field within the Packet Driver message
- a User Datagram Protocol (UDP) header field within the Packet Driver message
- Data Mover Protocol fields within the Packet Driver message
- a Routing Header within the Packet Driver message
- a Function Type field within the Routing Header
- a Message Count field within the Routing Header
- a Data Length field within the Routing Header
- a Message Header within the Packet Driver message
- a Version field within the Message Header
- a Length of Header field within the Message Header
- a Message Function Type field within the Message Header
- a Message Flag field within the Message Header

- a Protocol Type field within the Message Header
- a Sequence Number field within the Message Header
- a Source IP address field within the Message Header
- a Destination IP address field within the Message Header
- a Source port number field within the Message Header
- a Destination port number field within the Message Header
- a Length of data field within the Message Header
- a Status field within the Message Header
- the data from an intercepted packet within the Packet Driver message

A data communications system comprising one or more sets of peer-to-peer TCP/IP applications, a method of providing efficient and reliable delivery of the aggregated packet data over a communications path.

A data communications system comprising one or more sets of peer-to-peer TCP/IP applications, a method of separating the aggregated packet data following delivery over a communications path for subsequent delivery to the end-user networking applications.

Although the embodiment of the present invention exists as independent network devices, the present invention is not limited to this implementation. One who is skilled in the art of data communications and computer technology can implement the present invention in other ways including adapting it for use on other operating systems or computer platforms.

Brief Description of the Drawings

FIGURE 1A illustrates the placement of Transport Protocol Optimizers (TPO) in an end-to-end configuration.

FIGURE 1B illustrates an alternative implementation of the Transport Protocol Optimizer (TPO) in an end-to-end configuration, in which the Transport Protocol Optimizer is implemented on the host application servers.

FIGURE 1C illustrates an alternative implementation of the Transport Protocol Optimizer (TPO) in an end-to-end configuration, in which the Transport Protocol Optimizer is implemented in a network component.

FIGURE 2 illustrates the content of a TPO message.

FIGURE 3A illustrates the process of using a TPO communication link by two peer TCP/IP applications during the application connection sequence.

FIGURE 3B illustrates the process of using a TPO communication link by two peer TCP/IP applications after the application connection sequence has completed.

FIGURE 3C illustrates the process of using a TPO communication link by two peer UPD applications.

FIGURE 3D illustrates the process of using a TPO communication link by two peer ICMP applications.

FIGURE 4 illustrates the content of the Data Mover Network Request Block (NRB), which serves as the interface structure between the Packet Driver component and the Data Mover component.

FIGURE 5 illustrates the content of the Data Mover Driver portion of a TPO message.

FIGURE 6A illustrates the content of the Data Mover Network portion of a TPO message.

FIGURE 6B illustrates the content of the Data Mover Network Connect and Confirm subfields located in the Network portion of a TPO message.

FIGURE 6C illustrates the content of the Physical Address Map list.

FIGURE 6D illustrates the content of a Physical Address Map entry.

FIGURE 6E illustrates the content of a Physical Address Map route.

FIGURE 7A illustrates the content of the Data Mover Transport Base subfield located in the Transport portion of a TPO message.

FIGURE 7B illustrates the content of the Data Mover Transport Connect subfield located in the Transport portion of a TPO message.

FIGURE 7C illustrates the content of the Data Mover Transport Data subfield located in the Transport portion of a TPO message.

FIGURE 7D illustrates the content of the Data Mover Transport Acknowledgement subfield located in the Transport portion of a TPO message.

FIGURE 7E illustrates the content of the Data Mover Transport Disconnect subfield located in the Transport portion of a TPO message.

FIGURE 7F illustrates the process of establishing a Transport-Level communication link between two TPO units.

FIGURE 7G illustrates the process of establishing a new Transport-Level communication link between two TPO units following a communication path outage.

FIGURE 8A illustrates the content of the Data Mover Session Manager Base subfield located in the Session portion of a TPO message.

FIGURE 8B illustrates the content of the Data Mover Session Manager Connect subfield located in the Session portion of a TPO message.

FIGURE 8C illustrates the content of the Data Mover Session Manager Confirm subfield located in the Session portion of a TPO message.

FIGURE 8D illustrates the content of the Data Mover Session Service Base subfield located in the Session portion of a TPO message.

FIGURE 8E illustrates the content of the Data Mover Session Service Null subfield located in the Session portion of a TPO message.

FIGURE 8F illustrates the process of establishing a Session-Level communication link between two TPO units.

FIGURE 9 illustrates the content of the Packet Driver Application Level data located in the Application portion of a TPO message.

FIGURE 9A illustrates the content of the Packet Driver Application Level Routing Header data located in the Application portion of a TPO message.

FIGURE 9B illustrates the content of the Packet Driver Application Level Message Header data located in the Application portion of a TPO message.

Detailed Description of the Invention

The present invention provides a protocol and method for enhancing the throughput of Transmission Control Protocol / Internet Protocol (TCP/IP) applications. The present invention, called Transport Protocol Optimizer (TPO), provides a mechanism to maximize the performance of TCP/IP applications, particularly when large amounts of data are sent over extended distances. It achieves this goal by intercepting IP packets on a Local Area Network (LAN) and placing them in buffers for subsequent delivery over an already established UDP communication path between two Transport Protocol Optimizers. If the local application uses the Transport Control Protocol (TCP), that connection is terminated by the TPO that resides on the same network segment as the application server. The packet interception and UDP delivery is performed without the originating TCP/IP application being aware of this redirection.

For maximum optimization, the TPO connection normally exists over a Wide Area Network (WAN), although this is not a requirement. A TPO connection may exist over any type of IP data communications network.

The TPO protocol provides a method for successful delivery of the intercepted IP packet data over a UDP communication path. Upon reaching the destination TPO, the intercepted packets are removed from the UDP buffer and delivered to the destination server. If the intercepted application comprises a TCP connection, then an independent TCP connection is established between the destination TPO and the application executing on the destination server.

The present invention is capable of intercepting selected IP packets based on the following quintuple:

1. source IP address
2. source port number
3. destination IP address
4. destination port number
5. protocol type.

The present invention is limited to intercepting IP packets comprising either the TCP, UDP, or ICMP protocols. However, it should be noted that the concept of the present invention is not limited to these interception types. It is anticipated that a person who is skilled in data communication software can extend the method of the present invention to include other TCP/IP protocols.

In FIGURES 1A, 1B, and 1C, the terms Network Router and Network Switch are used to refer to standard networking components. They are typically an integral part of all networking infrastructures, and are readily available from many different network component vendors. They are outside the scope of the present invention.

FIGURE 1A represents an embodiment of the present invention. Application Server 10 is a server on which a TCP/IP application 11 executes. It communicates with another TCP/IP application 51 that executes on Application Server 50. Network Switch 20 provides connections to Application Server 10 with Cable 21, to Network Router 30 with Cable 22, and to Transport Protocol Optimizer 40 with Cable 24.

Collectively, this portion of the network comprises Local Area Network 1. Network Switch 60 provides connections to Application Server 50 with Cable 61, to Network Router 70 with Cable 62, and to Transport Protocol Optimizer 80 with Cable 64. Collectively, this portion of the network comprises Local Area Network 2. Local Area Network 1 is interconnected to Local Area Network 2 by the interconnection of Network Router 30 to the Wide Area Network (WAN) 90 with Cable 23, and with Network Router 70 to the Wide Area Network (WAN) 90 with Cable 63.

Referring to FIGURE 1A, the data flow through the TPO configuration is as follows:

Using standard IP routing mechanisms, data that is generated by Application 11 flows from Application Server 10, then to Network Switch 20, then to TPO 40, then back to Network Switch 20, then to Network Router 30, then to WAN 100, then to Network Router 70, then to Network Switch 60, then to TPO 80, then to Network Switch 60, then to Application Server 50, then to Application 51. Data generated by Application 51 flows in the reverse fashion back to Application 11.

Although the embodiment of the present invention is implemented as independent host servers (as represented in FIGURE 1A), the present invention is not limited to this embodiment. FIGURES 1B and 1C represent a conceptual embodiment of the present invention in other forms.

In FIGURE 1B, Application Server 110 is a server on which a TCP/IP application 111 executes, and on which the Transport Protocol Optimizer is implemented. It communicates with another TCP/IP application 151 that executes on Application Server 150, on which the Transport Protocol Optimizer is implemented. Network Switch 120 provides connections to Application Server 110 with Cable 121, and to Network Router 130 with Cable 122. Collectively, this portion of the network comprises Local Area Network 1. Network Switch 160 provides connections to Application Server 150 with Cable 161, and to Network Router 170 with Cable 162. Collectively, this portion of the network comprises Local Area Network 2. Local Area Network 1 is interconnected to Local Area Network 2 by the interconnection of Network Router 130 to the Wide Area Network (WAN) 190 with Cable 123, and with Network Router 170 to the Wide Area Network (WAN) 190 with Cable 163. It is anticipated that a person who is skilled in writing data communication software can implement the Transport Protocol Optimizer in the embodiment represented in FIGURE 1B.

In FIGURE 1C, Application Server 210 is a server on which a TCP/IP application 211 executes. It communicates with another TCP/IP application 251 that executes on Application Server 250. The Transport Protocol Optimizer is integrated into Network Switch 220 and Network Switch 260, or it is integrated onto an internal server (e.g. a Blade component) in each of those Network Switches. Network Switch 220 provides connections to Application Server 210 with Cable 121, and to Network Router 230 with Cable 222. Collectively, this portion of the network comprises Local Area Network 1. Network Switch 260 provides connections to Application Server 250 with Cable 261, and to Network Router 270 with Cable 262. Collectively, this portion of the network comprises Local Area Network 2. Local Area Network 1 is interconnected to Local Area Network 2 by the interconnection of Network Router 230 to the Wide Area Network (WAN) 290 with Cable 223, and with Network Router 270 to the Wide Area Network (WAN) 290 with Cable 163. It is anticipated that a person who is skilled in writing data communication software can implement the Transport Protocol Optimizer in the embodiment represented in FIGURE 1C.

The Transport Protocol Optimizer consists of three components:
1. Packet Interceptor
2. Packet Driver
3. Data Mover

The Packet Interceptor is a component that is responsible for intercepting Internet Protocol (IP) packets that are generated from user TCP/IP applications and redirecting them to the Transport Protocol Optimizer.

The Packet Driver is a component that is responsible for aggregating the intercepted IP packet data into buffers, for subsequent delivery over a communication link to a peer Packet Driver component, for subsequent delivery to the peer user application.

The Data Mover is a component that provides the reliable data delivery mechanism over a network communication link.

FIGURE 2 illustrates the content of TPO data units. User Datagram Protocol (UDP) is used as the mechanism for delivering the intercepted IP packet data over the Internet Protocol (IP) network to the destination Transport Protocol Optimizer

TPO data units are comprised of an Internet Protocol (IP) header field 300, a User Datagram Protocol (UDP) header field 301, and UDP payload data. The UDP payload data consists of Data Mover Driver protocol data 302, Data Mover Network protocol data 303, Data Mover Transport protocol data 304, Data Mover Session protocol data 305, Packet Driver Application protocol data 306, and intercepted IP packet data 307. The data associated with each protocol layer is meaningful to the corresponding peer layer component in the destination Transport Protocol Optimizer.

The present invention therefore contains a method of intercepting Internet Protocol (IP) packets generated from TCP/IP applications, and forming TPO data units for delivery to the remote Transport Protocol Optimizer, with subsequent delivery of the intercepted IP packet data to the remote peer TCP/IP application.

Although the embodiment of the present invention is limited to IPV4 (Internet Protocol Version 4) networks, the present invention should not be limited to this embodiment. It is anticipated that a person who is skilled in writing data communication software can implement the Transport Protocol Optimizer for IPV6 (Internet Protocol Version 6) networks.

Packet Interceptor

Paramount to achieving a method of optimizing TCP/IP applications, is the ability to transparently intercept Internet Protocol (IP) packets that are generated by the applications for which optimization is desired. In the present invention, this interception occurs by using a series of hooks located in various points in the Linux TCP/IP protocol stack. The definition of packet addresses (quintuples) for which packets are to be intercepted is defined in a local configuration file.

The NF_IP_PRE_ROUTING hook is used to examine the headers in IP packets as they arrive in the IP stack from the network, and look for matching quintuples (source IP [port], destination IP [port], protocol) for which optimization is desired. If optimization is defined for the quintuple defined in the IP packet, the packet is redirected to the local Transport Protocol Optimizer by modifying the destination IP address in the IP packet to be that of the local network interface, and by modifying the destination port number in the IP packet to be that of the Transport Protocol Optimizer application.

The NF_IP_LOCAL_OUT hook is used to examine the headers in IP packets as they leave the IP stack onto the network, and look for matching quintuples (source IP [port], destination IP [port], protocol) for which optimization is desired. If optimization is defined for the quintuple defined in the IP packet, the packet is modified so that the source IP address and source port number in the IP packet contains the IP address and port number of the originating application server.

As packets are intercepted for a given quintuple, a process is used on each side of the Transport Protocol Optimizer to serve as the local endpoint for each stream of packets. This process is referred to as an Edge process. There is a separate Edge process created that serves as the endpoint for each unique quintuple for packets generated by applications using the TCP/IP or UDP protocols. There is a separate Edge process created that serves as the endpoint for all packets generated by applications using the ICMP protocol.

FIGURE 3A illustrates the process flow that occurs during the establishment of a communication link between two peer TCP/IP applications when using a TPO communication link. Refer to FIGURE 1A for the identification of Application Server 10, Application Server 50, TPO 40, TPO 80, Application 11, and Application 51.

A 400 TCP/IP Connect request is issued by TCP/IP Application 11, running on Application Server 10, requesting a connection to TCP/IP Application 51, which is running on Application Server 50. The Connect request is directed to TPO 40 by means of standard IP Routing mechanisms, which results in 401 the IP packets entering the IP stack of TPO 40. The 402 NF_IP_PRE_ROUTING hook is entered, and 403 determines if this packet should be selected, based on a set of predefined filtering rules for matching source IP address, source IP port, destination IP address, destination IP port, and protocol type. If the packet should not be intercepted, then 404 no changes are made to the packet addressing, and the packet continues through the stack and follows the routing rules in effect for that IP destination address. If the packet should be intercepted, then 405 the packet destination IP address is changed to the local network interface on TPO 40, and the destination port number is changed to the local port on which TPO 40 is listening. The 406 application Connect Request is then queued while 407 the TPO 40 places the Application Connect Request IP packet, along with a Message Header, into a buffer to be sent to TPO 80 over the Data Mover UDP communication path. The Message Header consists of the original source IP address, source IP port, destination IP address, destination IP port, and protocol type. In this embodiment of the present invention, a 408 Edge process is created to serve as the local connection endpoint for TCP/IP Application 11. When the Connect Request is sent to TPO 80, the 409 NF_IP_LOCAL_OUT hook is entered. However, at this hook point, 410 nothing needs to be modified in the packet, since the IP addressing correctly identifies TPO 40 and TPO 80 as the source and destination endpoints. The 411 packets leave the stack and go onto the network, at which time 412 this process waits for a response from TPO 80.

When 419 TPO 80 receives the Connect Request, the 420 IP packets enter the IP stack from the network. The 421 NF_IP_PRE_ROUTING hook is entered. At this hook point, the 422 addresses in the IP packet do not need to be modified, since the IP addressing correctly identifies TPO 40 as the source and TPO 80 as the destination. The 423 Edge process is created, and a port number is assigned for the local endpoint on TPS 80. The Edge process serves as the local endpoint for the TCP/IP connection between TPO 80 and Application 51. The 424 Edge process issues the TCP/IP Connect request to Application 51, using the destination IP address and destination port number contained in the associated Message Header. The 425 NF_IP_LOCAL_OUT hook is entered, which then 426 modifies the source IP address to be that of Application Server 10, and modifies the source port number to be that of Application 11. The 427 packet leaves the stack and goes onto the network, at which time 428 this process then waits for the Connection Response from Application 51. When the 429 Connection Response is received, the 430 IP packets enter the IP stack from the network. The 431 NF_IP_PRE_ROUTING hook is entered, and 432 determines if this packet should be selected, based on a set of predefined filtering rules for matching source IP address, source IP port, destination IP address, destination IP port, and protocol type. If the packet should not be intercepted, then 433 no changes are made to the packet addressing, and the packet continues through the stack and follows the routing rules in effect for that IP destination address. If the packet should be intercepted, then 434 the packet destination IP address is changed to the local network interface on TPO 80, and the destination port number is changed to the local port on which TPO 80 is listening. Since this packet represents the connection response from Application Server 50, responding to the intercepted connection request from Application Server 10, this packet will be intercepted, since its source IP address would be Application Server 50, and its destination IP address would be Application Server 10. The 435 connection result is returned to TPO 40. The 436 NF_IP_LOCAL_OUT hook is then entered. At this hook point, 437 nothing needs to be modified in the packet, since the IP addressing correctly identifies TPO 80 as the source and TPO 40 as the destination.

When the 440 Connect Response is received from the remote TPO, the 441 IP packets enter the stack from the network. The 442 NP_IF_PRE_ROUTING hook is again entered. At this hook point, 443 nothing needs to be modified in the packet, since the IP addressing correctly identifies TPO 80 as the source and TPO 40 as the destination. If 444 the connect request was not successful on the remote TPO 80, then 445 a predefined invalid port number is set in the queued Connect Request, which will cause the local connection attempt to fail. If the connect request was successful on the remote TPO 80, then 446 the predefined port number of the local TPO 40 is set in the queued Connect Request, which will cause the local connection attempt to succeed. In either case, the 447 previously queued application Connect Request is requeued to attempt to establish the connection. TCP/IP will then 448 process the local Connect Request on TPO 40. If the valid port number was set in the Connect Request (meaning the connection was successfully established on the peer TPO 80), the connection to the local TPO Edge process will be accepted. If the invalid port

number was set in the Connect Request (meaning the connection was not established on the peer TPO 80), the connection to the local TPO Edge process will fail. In either case, there is an IP packet generated that reflects the acceptance or failure of the connection request. The 449 NF_IP_LOCAL_OUT hook is then entered for this packet, which then 450 modifies the source IP address to be that of Application Server 50, and modifies the source port number to be that of Application 51. The 451 packet leaves the stack and goes onto the network to the source application, at which time the 452 connection request is complete, either successfully or unsuccessfully.

FIGURE 3B illustrates the process flow that occurs after the establishment of a communication link between two peer TCP/IP applications when sending data over a TPO communication link. Refer to FIGURE 1A for the identification of Application Server 10, Application Server 50, TPO 40, TPO 80, Application 11, and Application 51.

A 460 TCP/IP Write request is issued by TCP/IP Application 11, running on Application Server 10, sending data to TCP/IP Application 51, which is running on Application Server 50. The Write request is directed to TPO 40 by means of standard IP Routing mechanisms, which results in 461 the IP packets entering the IP stack of TPO 40. The 462 NF_IP_PRE_ROUTING hook is entered, and 463 determines if this packet should be selected, based on a set of predefined filtering rules for matching source IP address, source IP port, destination IP address, destination IP port, and protocol type. If the packet should not be intercepted, then 464 no changes are made to the packet addressing, and the packet continues through the stack and follows the routing rules in effect for that IP destination address. If the packet should be intercepted, then 465 the packet destination IP address is changed to be that of the local network interface on TPO 40, and the port number is changed to be the local port on TPO 40 on which the Edge process for this connection is reading (which was created during the Connection sequence). The 466 Edge process then reads the intercepted packet, 467 queues the packet, along with a Message Header, to the Packet Driver for subsequent insertion into the Packet Driver buffer. When the 468 buffer is full, or as soon as Data Mover resources are available to send the buffer, the Packet Driver sends the buffer to the remote TPO over the Data Mover UDP communication path. The Message Header consists of the original source IP address, source IP port, destination IP address, destination IP port, and protocol type. When the UDP buffer is sent to TPO 80, the 469 NF_IP_LOCAL_OUT hook is entered for each IP packet. However, at this hook point, 470 nothing needs to be modified in the packet, since the IP addressing correctly identifies TPO 40 and TPO 80 as the source and destination endpoints. The 471 packets leave the stack and go onto the network, at which time 472 TPO 40 is done processing the Write request.

Asynchronous to 466, the 475 TCP/IP in TPO 40 will generate an ACKnowledgement message. The 476 NF_IP_LOCAL_OUT hook is entered for this packet, which then 477 modifies the source IP address to be Application Server 50, and modifies the source port number to be Application 51. The 471 packet leaves the stack and goes onto the network.

When 480 TPO 80 receives the sent buffer, the 481 IP packets enter the IP stack from the network. The 482 NF_IP_PRE_ROUTING hook is entered. At this hook point, the 483 addresses in the IP packet do not need to be modified, since the IP addressing correctly identifies TPO 40 as the source and TPO 80 as the destination. The 484 packets are removed from the incoming buffer by the Packet Driver, and queued to the correct Edge process, based on the quintuple identified in the Message Header associated with each packet. The 485 Edge process, serving as the local endpoint of the connection, issues the TCP/IP Write request to Application 51, using the destination IP address and port number contained in the associated Message Header. The 486 NF_IP_LOCAL_OUT hook is entered, which then 487 modifies the source IP address to be Application Server 10, and modifies the source port number to be Application 11. The 488 packets leave the stack and go onto the network, at which time 489 TPO 80 is done processing the Write request.

Asynchronous to 485 (but after 485), the 490 TCP/IP in Application Server 50 will generate an ACKnowledgement message destined for TPO 80. When TPO 80 receives the ACK, the 491 packet enters the IP stack from the network. The 492 NF_IP_PRE_ROUTING hook is entered, which then 493 modifies

the destination IP address and port number to be that of TPO 80. The 493 packet goes up the stack for processing by TPO 80.

FIGURE 3C illustrates the process flow that occurs after the establishment of a communication link between two peer UDP applications when sending data over a TPO communication link. Refer to FIGURE 1A for the identification of Application Server 10, Application Server 50, TPO 40, TPO 80, Application 11, and Application 51.

A 500 UDP Write request is issued by UDP Application 11, running on Application Server 10, sending data to UDP Application 51, which is running on Application Server 50. The Write request is directed to TPO 40 by means of standard IP Routing mechanisms, which results in 501 the IP packets entering the IP stack of TPO 40. The 502 NF_IP_PRE_ROUTING hook is entered, and 503 determines if this packet should be selected, based on a set of predefined filtering rules for matching source IP address, source IP port, destination IP address, destination IP port, and protocol type. If the packet should not be intercepted, then 504 no changes are made to the packet addressing, and the packet continues through the stack and follows the routing rules in effect for that IP destination address. If the packet should be intercepted, then 505 a port number is assigned and the Edge process is created to serve as the local endpoint if this is the first occurrence of a UDP packet for this quintuple. Then 506 the packet destination IP address is changed to be that of the local network interface on TPO 40, and the port number is changed to be the local port on TPO 40 on which the Edge process for this quintuple is listening. The 507 Edge process then reads the intercepted packet, and 508 queues the packet, along with a Message Header, to the Packet Driver for subsequent insertion into the Data Mover buffer. When the 509 buffer is full, or as soon as Data Mover resources are available to send the buffer, the Packet Driver sends the buffer to the remote TPO over the Data Mover UDP communication path. The Message Header consists of the original source IP address, source IP port, destination IP address, destination IP port, and protocol type. When the UDP buffer is sent to TPO 80, the 510 NF_IP_LOCAL_OUT hook is entered for each IP packet. However, at this hook point, 511 nothing needs to be modified in the packet, since the IP addressing correctly identifies TPO 40 and TPO 80 as the source and destination endpoints. The 512 packets leave the stack and go onto the network, at which time 513 TPO 40 is done processing the Write request.

When 520 TPO 80 receives the sent buffer, the 521 IP packets enter the IP stack from the network. The 522 NF_IP_PRE_ROUTING hook is entered. At this hook point, the 523 addresses in the IP packet do not need to be modified, since the IP addressing correctly identifies TPO 40 as the source and TPO 80 as the destination. If this is the first occurrence of a UDP packet for this quintuple, then 524 a port number is assigned and the Edge process is created to serve as the local endpoint for this quintuple. The 525 packets are removed from the incoming buffer by the Packet Driver, and passed to the assigned Edge process, based on the quintuple identified in the Message Header associated with each packet.

The 526 Edge process, serving as the local endpoint of the connection, issues the UDP Write request to Application 51, using the destination IP address and port number contained in the associated Message Header. The 527 NF_IP_LOCAL_OUT hook is entered, which then 528 modifies the source IP address to be that of Application Server 10, and modifies the source port number to be that of Application 11. The 529 packets leave the stack and go onto the network, at which time 530 TPO 80 is done processing the Write request.

FIGURE 3D illustrates the process flow that occurs after the establishment of a communication link between two peer ICMP applications when sending data over a TPO communication link. Refer to FIGURE 1A for the identification of Application Server 10, Application Server 50, TPO 40, TPO 80, Application 11, and Application 51.

The Edge processes are created and port numbers assigned during TPO initialization to handle all ICMP requests. These Edge processes serve as the local endpoints for ICMP requests issued by Application 11 running on Application Server 10, and by Application 51 running on Application Server 50.

A 540 ICMP Write request is issued by ICMP Application 11, running on Application Server 10, sending data to ICMP Application 51, which is running on Application Server 50. The Write request is directed to TPO 40 by means of standard IP Routing mechanisms, which results in 541 the IP packets entering the IP stack of TPO 40. The 542 NF_IP_PRE_ROUTING hook is entered, and 543 determines if this packet should be selected, based on a set of predefined filtering rules for matching source IP address, source IP port, destination IP address, destination IP port, and protocol type. If the packet should not be intercepted, then 544 no changes are made to the packet addressing, and the packet continues through the stack and follows the routing rules in effect for that IP destination address. If the packet should be intercepted, the 545 packet destination IP address is changed to be that of the local network interface on TPO 40, and the port number is changed to be the local port on TPO 40 on which the ICMP Edge process is listening. The 546 Edge process then reads the intercepted packet, and 547 queues the packet, along with a Message Header, to the Packet Driver for subsequent insertion into the Data Mover buffer. When the 548 buffer is full, or as soon as Data Mover resources are available to send the buffer, the Packet Driver sends the buffer to the remote TPO over the Data Mover UDP communication path. The Message Header consists of the original source IP address, source IP port, destination IP address, destination IP port, and protocol type. When the UDP buffer is sent to TPO 80, the 549 NF_IP_LOCAL_OUT hook is entered for each IP packet. However, at this hook point, 550 nothing needs to be modified in the packet, since the IP addressing correctly identifies TPO 40 and TPO 80 as the source and destination endpoints. The 551 packets leave the stack and go onto the network, at which time 552 TPO 40 is done processing the Write request.

When 560 TPO 80 receives the sent buffer, the 561 IP packets enter the IP stack from the network. The 562 NF_IP_PRE_ROUTING hook is entered. At this hook point, the 563 addresses in the IP packet do not need to be modified, since the IP addressing correctly identifies TPO 40 as the source and TPO 80 as the destination. The 564 packets are removed from the incoming buffer by the Packet Driver, and passed to the ICMP Edge process.

The 565 Edge process, serving as the local endpoint of the connection, issues the ICMP Write request to Application 51, using the destination IP address and port number contained in the associated Message Header. The 566 NF_IP_LOCAL_OUT hook is entered, which then 567 modifies the source IP address to be that of Application Server 10, and modifies the source port number to be that of Application 11. The 568 packets leave the stack and go onto the network, at which time 569 TPO 80 is done processing the Write request.

Data Mover

The Data Mover is the component that is responsible for providing the data delivery mechanism to the remote Transport Protocol Optimizer. Since UDP is used as the underlying IP delivery mechanism, the Data Mover is responsible for handling all flow control, retransmissions, and ordered delivery of data to the end-user application.

When viewed from the perspective of the Open System Interconnection (OSI) Reference Model seven layer communication stack, the Data Mover can be thought of as providing unique Layer 3 (Driver), Layer 4 (Transport), Layer 5 (Session), and Layer 6 (Application) services.

Driver Layer

The driver layer interfaces with the IP stack by using UDP socket calls, and is responsible for sending and receiving network messages in and out of a Network Interface Card (NIC).

Network Layer

The principal function of the Network Layer is to establish a path between two Transport Protocol Optimizers, construct the messages that are needed to send data between them, and remove the routing information before presenting the incoming data to the Network caller (i.e. Transport Layer). The Network

layer also performs all the multiplexing for the Transport Protocol Optimizers - the process of sorting out incoming messages by unique connection identifier, and passing the data to the higher level caller.

Transport Layer
The main responsibility of the Transport layer is to provide guaranteed, correct data delivery to the remote Transport Protocol Optimizer, and is designed to function in environments where very long propagation delays, extremely high speeds, and high error rates may all be encountered.

To accomplish this, the Data Mover provides:

- The ability to negotiate transmission block sizes between both sides of the connection.
- The ability to segment an arbitrarily large block of application data into messages suitable for transmission, and to reassemble the data block for the receiving application.
- An acknowlegement scheme so that messages lost in transit can be retransmitted.
- A flow control facility so that the receiving Transport Protocol Optimizer is not overrun with data.
- Alternate path retry capability, where alternate routes to a destination will be tried if a primary one fails.

Transport will deliver data in the correct order at the pace no greater than that desired by the receiver. It will either deliver the data successfully or it will report that communications with the remote transport have been lost over all possible paths to the remote host. In that event, the status of data not explicitly acknowledged by the remote application is not known.

Session Layer
Session service is fairly similar to the services offered by the Transport Layer. The major difference is that the Session Layer provides the ability to address a remote application by two character string names: a "host" name (i.e. the hostname of the remote Transport Protocol Optimizer) and an "application" name (i.e. the name of the remote Packet Driver).

Application Layer (Packet Driver)
The Application Layer services are provided by a Packet Driver component, which is responsible for aggregating intercepted packets, along with the packet routing information, into a TPO buffer. When either the buffer is full, or TPO Session Services are available, the local Packet Driver uses Session Services to send this buffer over the TPO connection to the peer Packet Driver component on the other side of the connection.

Each TPO buffer can contain intercepted IP packet data from more than one application server, and can contain intercepted IP packet data that is destined to more than one application server. In other words, all of the data for the intercepted IP packets in a given TPO buffer may have different source IP addresses and/or different destination IP addresses. It is up to the receiving TPO Packet Driver to process the routing information received with each packet, and deliver the IP packets to the correct destination.

DATA MOVER NETWORK REQUEST BLOCK

Communication between the Data Mover component and the Packet Driver component takes place by sharing a data structure, referred to as the Data Mover Network Request Block (NRB). This structure contains fields that are examined and updated by both sides of the connection.

Data Mover calls consist of service requests to establish a connection, transfer data, and terminate a connection. Not all calls are available for all levels of service. The types of service calls available are

shown below:

Service	Levels available
CONNECT	Session, Transport, Network, Driver
OFFER	Session, Transport, Network
CONFIRM	Session, Transport, Network
WRITE	Session, Transport, Network, Driver
READ	Session, Transport, Network, Driver
CLOSE	Session, Transport
DISCONNECT	Session, Transport, Network, Driver

CONNECT is used to establish a connection to the remote Packet Driver component through a matching OFFER. It allocates the resources needed to maintain a connection at the session level and all lower levels. Data associated with each level will be delivered to the matching layer on the other end along with the Connect Indication.

OFFER is used when a Packet Driver wishes to be connected to by a peer Packet Driver. It allocates the connection resources needed to receive connection data from the network. Successful completion of the OFFER provides a Connect Indication as well as the data sent by the CONNECT.

CONFIRM is issued upon completion of an OFFER by the Packet Driver that accepted the connection. It may provide data that will be sent back to the Connecting Packet Driver. Upon receipt of the Confirm by the connecting side, the connection is considered complete.

WRITE is the normal means of sending data to the other side. A WRITE request completes as soon as the local Data Mover has accepted responsibility for the data.

READ is a signal to the Data Mover that the Packet Driver is prepared to accept further data or indications from the Data Mover. The READ mechanism is used by the Data Mover to allow the Packet Driver to accept data from the network at a sustainable rate.

CLOSE is used to gracefully terminate the connection. When one Packet Driver issues a CLOSE, no further transmissions may take place from that Packet Driver. When the second side issues a CLOSE, the connection is terminated.

DISCONNECT is used to abruptly terminate a connection. The other side receives a Disconnect Indication and any data in progress will be lost.

The Packet Driver component interfaces to the Data Mover component at the Session level.

DATA MOVER NETWORK REQUEST BLOCK FORMAT

The Data Mover Network Request Block (NRB) provides the interface structure between the Packet Driver

component and the Data Mover component.

FIGURE 4 illustrates the content of the Data Mover Network Request Block.

NRBSTAT: 1300 contains a status code for the request issued by the Packet Driver. If the request is currently in progress, the word contains a negative value. If the request completed successfully, then this word contains a zero. If the Data Mover detected an error, it will appear as a binary value in this field.

NRBIND: 1301 indicates the type of data received in response to a READ or OFFER request. If any of these read type requests are issued, NRBIND receives a nonzero value. If a write type request (WRITE, CONNECT, CONFIRM, CLOSE, or DISCONNECT) is issued, the returned value of NRBIND is usually zero. If an error is returned to the write type request that means the connection is broken or was never established, then a Disconnect Indication (6) will be set in NRBIND.

The values returned in NRBIND are defined as follows:

(1) - Connect Indication
(2) - Confirm Indication
(3) - Normal data Indication
(5) - Close indication
(6) - Disconnect indication

If an operation did not complete successfully, then NRBSTAT will be set to a positive, nonzero value. If NRBSTAT is nonzero, then NRBIND will have one of the following values:

- If the error results in the loss of the connection or the connection not being established in the first place, then a Disconnect Indication (6) will be in NRBIND.
- If the error means that the request could not be processed but the connection remains in effect, then NRBIND will be set to zero.
- If the data is "damaged" in input (user buffer too small, bad DATAMODE, etc.), then NRBIND will reflect the type of data received.

NRBLEN: 1302 specifies the number of words that are needed to contain the input or output data. For read operations, NRBLEN reflects the amount of data received from the transmitting side; for write operations, NRBLEN specifies the length of data to send.

NRBREQ: 1303 is the Request Code that is to be given to the Data Mover. This is a sixteen bit binary value that contains the type of request (example: SREAD) that the Data Mover is to perform. Another bit specifies whether the Data Mover is to automatically suspend program execution until the request has completed.

The values of NRBREQ are split into three fields as follows:

Option Flags	Service Level	Function

Option Flags refers to optional processing that the Data Mover will perform on the request. The binary format of this field is as follows:

0000 indicates that this request is to be WAITed upon, and that control will not return

to the Packet Driver until this request is complete.

1000 indicates that control is to be returned to the Packet Driver as soon as the Data Mover has internally queued the request.

Service level indicates if the request is a SESSION, TRANSPORT, DRIVER, or other type of request. Assigned values are:

0 - SESSION request
1 - TRANSPORT request
2 - NETWORK request
3 - DRIVER request

Function indicates the specific type of request to be issued. Because of the similar nature of the Data Mover request codes at each of the service levels, the total request is produced by adding the Function and service level - for example, SREAD is a 0 (Service level) plus a 2 (function) for a 0002 (hex) request code. Function assignments are as follows. All numbers are in hex.

01 — Connect (Valid for all levels)
02 — Confirm (Valid for all levels)
03 — Write (Valid at all levels)
04 — Expedited Write
05 — Close (Valid for Session & Transport)
06 — Disconnect (Valid for S, T, N and D levels)
07—7F — Reserved.
81 — Offer (Valid for S, T, and N levels)
82 — Read (Valid for all levels)
84—FF — Reserved.

NRBNREF: 1304 contains a 16 bit internal connection identifier that distinguishes this connection from all others maintained by the Data Mover. If a Session Connect or Offer is made, this value should initially be set to zero, and an identifier will be assigned by the Data Mover and returned to the Session caller.

NRBBUFA: 1305 contains the start of the data buffer to be used for either input or output requests. The Packet Driver supplies a valid buffer address before each input or output request. In the case of a write request, the contents of this buffer must be left unchanged until the associated Data Mover write type request has completed. If a read request is issued, then the contents of the buffer may be processed when the read request completes successfully.

NRBBUFL: 1306 For output requests, this field is ignored. For input requests, it specifies the maximum length of data that the Data Mover can store in the buffer.

NRBDMODE: 1307 Datamode is specified by the transmitting Packet Driver on any write request. It is always specified as a 16 bit quantity. Datamode is forwarded through all layers of the Data Mover. When the receiving side receives the data, the datamode specified by the transmitter is inserted into the NRB associated with the receiving READ request. Datamode supports three basic types of conversion options

Bit Stream - the bit pattern sent is precisely reproduced in the destination machine
Octet - a mode where 8 bit binary quantities are sent
Character - sending character information from one machine to another with a full range of character code conversion options.

The format of the datamode field is as follows:

Source Character Set	Destination Character Set

If both "source" and "destination" fields are zero (the default), then data is sent in bit stream mode.

If both fields contain a "1", then the data is sent in octet mode.

Values greater than one indicate an index that specifies a certain character set. The transmitter specifies the character code used by the information in the write buffer, and the character code in which the information should be received by the corresponding application. Current indices for character sets are:

2 - ASCII (8 bit)
3 - EBCDIC
4 - 7 - reserved

The Packet Driver transmits all buffers in bit mode.

NRBTIME: 1308 is used to specify the length of elapsed time that the associated read-type request is to remain in effect. If a time interval equal to the number of seconds in NRBTIME has elapsed and no data or connection information has arrived to satisfy the READ or OFFER, then the request will end with an error.

If the value in NRBTIME is zero, then the request will wait indefinitely.

CONNECTION NEGOTIATION PARAMETERS

The following parameters are associated with the session negotiation process. Information in these fields is updated by the Data mover as their values change. These fields are initially specified during the OFFER and CONNECT requests. When the OFFERing task receives the connect, the negotiated values are set in the OFFERed NRB. When the CONFIRM is sent, the negotiated values are set in the NRB associated with the READ of the CONFIRM information. Subsequent attempts to change these fields will have no effect.

NRBCLASS: 1309 specifies the type of protocol that will be used by the connection service. The values defined for class are as follows:

0 - use default class (4).
1 - reserved
2 - use Version 2 Data Mover protocol
4 - use Version 4 Data Mover protocol (standard)

The Packet Driver uses Version 4 Data Mover protocol.

NRBMXRAT: 1310 specifies the maximum data rate possible to the connection based on the Packet Driver's specification or the physical characteristics of the links.

Upon completion of the OFFER or READ following an CONNECT request, this field will have a nonzero value that contains the maximum throughput that is possible to the connection, based on the Packet Driver's original request and the characteristics of the communications link between the two.

NRBBLKI 1311 and **NRBBLKO 1312:** specifies the maximum size block of data that the application program expects to read and write during the coming connection. This parameter is provided with the

CONNECT and OFFER requests. During the protocol negotiation process, the BLKI of one side of the connection is compared with the BLKO (output maximum buffer size) specified at the other end, and the lesser of the two values will be returned in the two respective fields.

For the CONNECTing Packet Driver, the negotiated results will be returned in the NRB along with the CONFIRM data read following the CONNECT. The OFFERing Packet Driver will receive the negotiated values upon completion of the OFFER.

Two default options are available with these fields. If a zero is specified in either one of these fields, then the value used for negotiation will be an installation supplied default that is provided at Data Mover installation time. If the value in this field is the machine representation of -1, then the size used for negotiation will be the maximum size available for that installation, which is also a parameter specified at initialization time. Note that the value implied by the use of zero or -1 will be used for negotiation of the connection block sizes. The actual size negotiated will be supplied in these fields upon completion of the CONNECT and OFFER.

NRBPROTA 1313 and **NRBPROTL 1314:** These two fields permit the Packet Driver to provide Odata (octet protocol data) to the called layer of the Data Mover. When a write type command is issued, the Odata provided (if any) is added to the protocol message, and eventually delivered as Odata to the receiving Packet Driver's read-type command. As a result, this is a second buffer that is handled in a similar way to the data that is specified by NRBBUFA and NRBLEN. There are some distinct differences, however:

1. Odata is always sent and received in "octet mode," which means it will be represented by 8 bit binary quantities.

2. The maximum amount of Odata that may be sent is limited; this maximum is installation dependent and may typically be 256 bytes or less.

On a write type operation, no Odata will be sent if NRBPROTL is zero. If a nonzero length is specified, then the Odata will be transmitted along with the data, if present. When the read takes place, the data will be placed in the address specified by NRBPROTA and its incoming length will be set in NRBPROTL. In all cases, NRBPROTL contains the length of the Odata in octets.

NRBCONN1 1315 and **NRBCONN2 1316:** These fields are used to provide connection information when a CONNECT or OFFER request is issued. Their usage depends on the level of service requested. See the description of the services offered for more information.

DRIVER SERVICE INTERFACE

The driver layer interfaces with the IP stack by issuing UDP socket calls, and is used for sending and receiving network messages in and out of a Network Interface Card. Connecting to the driver establishes a "path" from the Data Mover to the network interface hardware.

Since the Packet Driver interfaces with the Data Mover at the Session Layer, the usage of the Driver Layer only occurs as the result of Packet Driver Session calls going down the protocol stack from Session to Transport, then to Network, and finally to the Driver Layer.

NETWORK SERVICE INTERFACE

The Network service interface is designed to supply a means of transmitting datagrams over any Internet Protocol (IP) media. It is responsible for establishing a communication path between two Transport Protocol Optimizers, constructing the messages that are needed to send data between them, and remove

routing information before presenting incoming data to the Network caller. The network layer also performs all the multiplexing for the Transport Protocol Optimizers - the process of sorting out incoming messages by unique identifier (N-ref), and passing the information to the Network caller (i.e. Transport level).

The Network Layer interfaces with the Driver Layer, which subsequently interfaces with the IP stack by issuing socket calls. Network service includes connection negotiation parameters, as well as a Read/Write and Disconnect facility. Network service is distinguished from the higher layers of communications software in that it is simply a delivery facility for "datagrams" and makes no promises of successful delivery of any data.

The Network user issues an N-OFFER or N-CONNECT, depending on whether it wishes to be the requestor or server in a connection. The N-OFFER user supplies data buffers to hold incoming data sent with the connecting party's N-CONNECT. The N-CONNECT user may specify data to be delivered to the OFFERing party. The N-CONNECT caller provides a data structure (the Physical Address Map) containing the particulars of the route to be followed to reach the OFFERing process. The Network layer issues whichever Driver network messages are needed to establish the N-connection and analyzes their response (if any). The Network layer then transmits the data provided.

When the N-OFFERing side receives an N-CONNECT indication, it responds with an N-CONFIRM request that completes the connection. This CONFIRM request may contain data that can be read by the connecting side.

An N-WRITE request results in the construction of a network message that is sent along the path established by the N-CONNECT. The Network caller only provides the data to be delivered; the packaging of the data into network messages is handled by the Network Layer. An N-WRITE request completes as soon as it is signalled complete by the Driver Layer. Errors detected by the driver will be reported as equivalent Network errors. Data delivery by the Network layer is not guaranteed; Network completes responsibility for transmissions as soon as it leaves the local network adapter.

An N-READ request is a request to accept data issued by a matching N-WRITE. Upon completion of the N-WRITE, buffers will be filled with the data provided by the N-WRITE caller. Removal of routing information and reassembly of the data as it arrives from the network is the responsibility of the caller. If a driver detected error occurs in an attempt to input data, the data will be forwarded to complete the Network read if the Network protocol of the incoming data is correct.

An N-DISCONNECT will cause the network resources acquired to maintain the connection to be freed. Any pending messages in the network will be lost. No indication is provided to the Network user on the other side - it is assumed that the corresponding user will issue its own N-DISCONNECT.

N-CONNECT - ESTABLISH NETWORK CONNECTION.

The N-CONNECT request is used to establish a path to another Transport Protocol Optimizer, to a process that has issued a corresponding N-OFFER request. Network service will issue whatever protocol messages are needed to establish the paths, circuits, or routes needed to communicate between the two Transport Protocol Optimizers. When the path is established, it delivers the data provided by the Packet Driver to the receiving Packet Driver, and the Network connection is then fully established.

The parameters of the N-CONNECT request are:

N-CONNECT	<NRBBUFA	- address of outgoing Pdata>
	<NRBBUFL	- length of outgoing Pdata>
	<NRBDMODE	- Datamode of Pdata>
	<NRBPROTA	- address of outgoing Odata>

	<NRBPROTL	- length of outgoing Odata>
	<NRBNREF	- connector's (local) N—ref>
	<NRBCONN1	- Physical Address Map of connection>
	<NRBCONN2	- offeror 's (remote) N—ref>
	<NRBBLKO	- Maximum transmit size of Pdata desired>
	<NRBBLKI	- Maximum transmit size of Odata desired>
	Options:	
	Wait	NoWait
Results:	NRBSTAT	- success/failure code
	NRBIND	- set to zero
	NRBMXRAT	- Max transmission speed of path

The connecting Network level user may specify two buffers of data to be sent to the receiving application. Odata is intended for protocol use and should be 256 octets or less in length. The length of the Pdata is dictated by the maximum size of a single transmission that is supported by the intermediate communications nodes.

A field in the NRBREQ word specifies if control should be returned to the caller as soon as any required message transmission is begun, or if control is to be returned only after the connect is complete.

The address of the process on the network that is to be connected to is provided in NRBCONN1. The NRBCONN1 field is the address of a data structure provided by the Network user called the "Physical Address Map" or PAM. This PAM is a description of all the physical paths to be followed to reach the desired destination, as well as the characteristics of the nodes and links between the two hosts. The PAM is discussed in detail later in this section.

The Network connection is matched with a specific other N-connection. through the use of two N-ref (connection identifier) fields. Each of these two fields is sixteen bits in length. Both the connecting parties place the connection identifier N-ref of their own local process in the NRBNREF field of the NRB. They place the N-ref of the remote party in the NRBCONN2 field of the NRB. Normally for the connection to take place, both parties must specify a matching, omplementary pair of N-ref's. The party issuing the N-OFFER has the option of specifying a zero in NRBCONN2, which implies that an N-Connect with any incoming remote N-ref is acceptable. In that case, the N-ref received will be placed in NRBCONN2 when the request completes. The value of the local connection N-ref in NRBNREF must be unique within the entire local Data Mover.

Two fields are used to inform the Network layer of the maximum amounts of Odata and Pdata that are to be used to send and receive data in this connection. These limits are dependent on many things: the buffer capacities generated in both the local and remote copies of the Data Mover, and the physical limitations of the media connecting the two hosts. When a following N-READ completes with a Confirm Indication, then these fields will have the actual limits for Odata and Pdata size in the connection sent to them. The Network Service will return the maximum that is available if the caller's size request is not available. It is then the responsibility of the caller to scale its buffer sizes downward accordingly.

The maximum size of Pdata is specified in addressable units; the maximum amount of Odata is specified in octets.

When invoked by an N-CONNECT, the Network layer will issue whatever lower level messages and functions are needed to establish a communication path through the intermediate nodes specified by the PAM. Once a clear path to the remote host exists, it will transmit the Pdata, Odata, and PAM provided by the user to the remote host. The N-CONNECT completes as soon as this data transmission completes locally. Delivery of the connect data is not guaranteed. To determine successful delivery of the connect

data, network applications should specify a timeout on the first N-READ following the N-CONNECT to verify that the Network application on the remote host has sent a response within a reasonable time. If no such response is received, then the local Network application should issue an N-DISCONNECT to free the connection resources followed by another N-CONNECT to the same destination. This process may continue until it is reasonably certain that the remote host or an intermediate node is "down" or does not have a matching N-OFFER request.

On completion, NRBMXRAT will contain the maximum data transmission rate that can be sustained through the connection. Generally this will be the speed of the slowest link. The speed is expressed in 1000's of bits per second.

N-OFFER - SOLICIT INCOMING N-CONNECTION.

The N-OFFER request results in the allocation of device resources needed to take part in an N-connection and causes the appropriate device driver to wait for incoming data. An N-OFFER completes when an N-CONNECT request from another host arrives, or the N-OFFER request times out. The parameters given for N-OFFER are:

N-OFFER	<NRBBUFA	- address for incoming data>
	<NRBBUFL	- max length of incoming Pdata>
	<NRBPROTA	- address to place incoming Odata>
	<NRBPROTL	- max length of incoming Odata>
	<NRBNREF	- local N-ref>
	<NRBTIME	- interval N-OFFER is outstanding>
	<NRBBLKO	- max transmit Pdata size desired>
	<NRBBLKI	- max transmit Odata size desired>
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- has Connect Indication
	NRBLEN	- length of Pdata received
	NRBDMODE	- Datamode of Pdata received
	NRBMXRAT	- Max transmission speed of path
	NRBBLKO	- Maximum send/receive Pdata size
	NRBBLKI	- Maximum send/receive Odata size
	NRBPROTL	- Length of Odata received
	NRBCONN1	- PAM information from connector
	NRBCONN2	- incoming N-ref from remote host

Since the Network layer performs message construction and disassembly the caller provides two buffers for input - a feature that will be continued in the higher Data Mover layers. If the length of either or both of the buffers specified is zero, then incoming data for that buffer type will be discarded and NRBSTAT will contain an error code reflecting the loss of data.

NRBNREF specifies local N-ref of the party issuing this connection. The remote party must provide an identical value in NRBCONN2 of the T-CONNECT request in order for a "match" to occur during the connection. Mechanisms outside of the network protocol must be used to agree on a suitable N-ref before the connection is established.

NRBTIME contains the number of seconds the N-OFFER is to remain in effect. If that real time interval elapses and no N-CONNECT message has been successfully received, then the N-OFFER will complete

with an error in NRBSTAT indicating that the offer has timed out.

The NRBBLKO and NRBBLKI fields are used to inform the Network layer of the maximum amounts of Odata and Pdata that are to be used to send and receive data in this connection. These limits are dependent on many. things: the buffer capacities generated in both the local and remote copies of the Data Mover, and the physical limitations of the media connecting the two hosts. When this N-OFFER completes with a Connect Indication, then these fields will have the actual limits for Odata and Pdata size in the connection sent to them. Unlike other layers of Data Mover service, the Network Service will return the maximum that is available if the caller's size request is not available. It is then the responsibility of the caller to scale its buffer sizes downward accordingly.

The maximum size of Pdata is specified in addressable units; the maximum amount of Odata is specified in octets.

When the N-OFFER successfully completes, NRBIND will contain the code for a Connect Indication to indicate the incoming event. The incoming Pdata will be pointed to by NRBBUFA. If move mode was selected with the original request, then the NRBBUFA must have been supplied by the Network application before the N-OFFER was issued. The Datamode of the incoming Pdata will have been placed in NRBDMODE.

On completion, NRBMXRAT will contain the maximum data transmission rate that can be sustained through the connection. Generally this will be the speed of the slowest link. The speed is expressed in 1000's of bits per second.

NRBBLKO contains the maximum amount of Pdata that may be sent or received on this Network connection. It is specified in addressable units.

NRBBLKI contains the maximum amount of Odata that may be sent or received on this Network connection. It is specified in octets.

On completion of the N-OFFER, the buffer pointed to by NRBCONN1 will contain the Physical Address Map used by the connecting process to complete the N-OFFER. This Physical Address Map is a complete description of a network path that will allow communications to take place between both parties. The Network layer will already have processed this incoming addressing information so that it will automatically return any subsequent N-WRITE's on this connection back to the originating party. This incoming PAM may be examined by the OFFERing application to determine the identity of the party that is contacting it. The size of this buffer should be 64 octets.

NRBCONN2 will contain the N-ref that will be used by the remote party. This value contains the value that the remote party specified in the NRBNREF field of its T-CONNECT. If a zero is specified in this field by the N-OFFER, then the first incoming N-CONNECT that specifies the correct local N-ref specified in NRBNREF will cause the N-OFFER to complete. In that case, the remote side's N-ref will be supplied in NRBCONN2.

N-CONFIRM - COMPLETE NETWORK CONNECTION

Whenever an N-OFFER or N-READ has successfully completed with a Connect Indication, an N-CONFIRM request must be issued to successfully complete the network connection. The Confirm request is very similar to the Write request documented below. If the local Network caller does not wish to communicate with the connecting party, then it may issue a N-DISCONNECT to avoid further conversation with that party.

Note that whenever data is read, a Connect Indication may arrive. Generally, this will be due to the fact that the other party detected a complete failure in the path between the two hosts and has issued another N-CONNECT over a different route. The party receiving the Connect Indication must respond with an N-ONFIRM before proceeding.

The parameters for N-CONFIRM are:

N—CONFIRM	<NRBBUFA	- address of outgoing Pdata>
	<NRBLEN	- length of outgoing Pdata>
	<NRBNREF	- local N-ref>
	<NRBDMODE	- Datamode of Pdata>
	<NRBPROTA	- address of outgoing Odata>
	<NRBPROTL	- length of outgoing Odata>
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- set to zero

As with N-CONNECT, two data buffers are supplied with N-CONFIRM. The Pdata is specified by NRBBUFA, NRBLEN, and NRBDMODE. The Odata to be sent is specified by NRBPROTA and NRBPROTL. The Network software will construct a message or messages to be introduced on the network and will transmit them. The request is complete when all data associated with the N-CONFIRM has been successfully introduced on the network.

N-WRITE - WRITE DATA ON NETWORK CONNECTION

After a Network connection is established (either side has either issued an N-CONFIRM or has received a Confirm Indication) either end of the connection may transmit data to one another through an N-WRITE request. The parameters for N-WRITE are:

N-WRITE	<NRBBUFA	- address of outgoing Pdata>
	<NRBLEN	- length of outgoing Pdata>
	<NRBNREF	- local N-ref>
	<NRBDMODE	- Datamode of Pdata>
	<NRBPROTA	- address of outgoing Odata>
	<NRBPROTL	- length of outgoing Odata>
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- set to zero

As with N-CONNECT, two data buffers are supplied with N-WRITE. The Pdata is specified by NRBBUFA, NRBLEN, and NRBDMODE. The Odata to be sent is specified by NRBPROTA and NRBPROTL. The Network software will construct a message or messages to be introduced on the network and will transmit them. The request is complete when all data associated with the N-WRITE has been successfully introduced on the network.

Any number of N-WRITE requests per connection may be outstanding at one time, although the Network layer may opt to force the application to wait for some to complete before accepting any more write requests.

N-READ - SOLICIT INDICATION FROM NETWORK

The N-READ request is used to signal the Network layer that the Network application is prepared to accept additional data (and indications) from the connection. It is the responsibility of the Network software to achieve maximum effectiveness of the use of the communications hardware, so the data may often be "read" (in the sense of being brought into computer memory) before the application requests it. The N-READ request also provides a mechanism for the Network layer to signal special events such as connection termination to the network application. For example, if the Network layer detects an unrecoverable communications loss in the connection, then it will schedule a Disconnect Indication to be "read" by the next N-READ after all correctly received data has been delivered.

The parameters used by N-READ are:

N-READ	<NRBBUFA - address for incoming data (move mode)>
	<NRBBUFL - max length of incoming Pdata>
	<NRBNREF - local N-ref>
	<NRBPROTA - address to place incoming Odata>
	<NRBPROTL - max length of incoming Odata>
	<NRBTIME - interval N-READ is outstanding>
	Options:
	Wait | NoWait
Results:	NRBSTAT - success/failure code
	NRBIND - incoming Indication type
	NRBLEN - length of Pdada received
	NRBDMODE - Datamode of Pdata received
	NRBPROTL - length of Odata received
	NRBCONN1 - New PAM if connect arrived

Basically, the application provides a set of data buffers and a timeout value for the N-READ request. If the Network layer has data ready to present to the Network application, then the request completes more or less immediately with the data supplied by the oldest received message for that data. Not all data that is written is necessarily received by the receiving application; data that does arrive is not guaranteed to arrive in the order of transmission.

If no data is currently available for the application, then the request will wait until data arrives. If no data arrives to satisfy the N-READ request in the time specified by NRBTIME, then the request will complete abnormally. If such a timeout occurs, the connection will remain complete; subsequent N-READs may be issued against the connection

Any number of N-READs may be outstanding against a single connection. If the total number of N-READs outstanding becomes excessive, the Network layer may force the application to wait until some of those read requests become complete. Any number of N-WRITE's may be issued while the N-READ remains outstanding, allowing full duplex communications to take place.

Since a disconnection is not signalled to both ends of a Network connection, it is possible for one party (generally the one that issued the original N-CONNECT) to issue. an N-DISCONNECT followed by an N-

CONNECT to the same N-ref. In that case, the inactive party will not see the disconnect indication; when the N-CONNECT arrives, the current N-READ will end with a Connect Indication in NRBIND, the new PAM used to reach this destination in a buffer pointed to by NRBCONN1, and connect data. This allows one party to establish a new communications path in the event of a failure without involving the other party. Note that when this subsequent connect takes place both the local and remote N-ref's must match on the receiving end; otherwise the data will be discarded.

If the Network layer detects loss of communications, then it will do so by completing an N-READ with a Disconnect Indication. All data successfully received will be used to satisfy previous N-READs before the Disconnect Indication is presented to the Network application. If the Network Layer is the one that originally issued the N-Connect, then it will make a best efforts attempt to free the network resources allocated to the connection before returning the Disconnect Indication to the application. After the Disconnect Indication is provided, all resources for the connection are freed and no further requests against that connection will be satisfied.

N-DISCONNECT - TERMINATE NETWORK CONNECTION

When a Network application determines that no further communications with the remote application is desired, it may terminate the connection with an N-DISCONNECT request. N—-DISCONNECT simply frees the resources that were originally allocated by the N-CONNECT/N-OFFER originally issued by each side.

If the issuer of an N-DISCONNECT was the application that originally issued the N-CONNECT, then the Network layer will free the network path that was allocated during the connection process. Any N-WRITE messages that have not yet been received by the remote host may be lost. After the network resources have been freed, the Network layer will free the local resources needed to maintain the connection; when this is done, the N- DISCONNECT will complete. Any further requests against that connection will be rejected. Any requests against that connection which were outstanding at the time the N-DISCONNECT was issued will complete with an error.

If the issuer of an N-DISCONNECT was the application that originally issued the N-OFFER, then the Network layer will only free the local resources needed to maintain the connection; the N-DISCONNECT will then complete. Any further requests against that connection will be rejected. Any requests against that connection which were outstanding at the time the N-DISCONNECT was issued will complete with an error.

The parameters for N-DISCONNECT are:

N-DISCONNECT		
	<NRBNREF	- local N—ref>
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	- success/failure code

N-WAIT - WAIT FOR NETWORK EVENT

Whenever a request for network service is issued, the application has two options - to give control to the Network layer until the particular request completes, or to have control returned as soon as the request is initiated. If control is returned to the application before the request completes, the Network application may then wait for that request or a series of requests to complete. This is done through the N-WAIT service.

The first option available to the user is that an N-WAIT may be issued on a list of outstanding NRB's that have requested Network service. When any one of the requests associated with those NRB's has completed (or any one of the requests completed before the N-wait was issued) then control will return to the application that issued the N-WAIT.

For those applications that are servicing complex real time events in addition to driving the Data Mover Network layer, the application may "poll" by issuing an N-WAIT with zero NRB's as a parameter. In that case, Network service will examine the state of any outstanding request, issue new I/O requests to service the network, and update any NRB's whose requests have completed. Thus, an application may issue this polling N-WAIT, have a prompt return of control from the Network layer, and then examine all its pending NRB's to see if they have completed

Once again, it should be emphasized that an NRB cannot be marked as complete until the Network layer obtains control in some way: an N-WAIT with that request in the NRB list, or polling via N- WAIT until the request is complete.

The format of the N-WAIT is:

N-WAIT <NRB-list>

The NRB-list is a table of addresses of NRB's to be waited on. Each entry in the list is a single word that contains the address of the first word of the NRB. The end of the list is indicated by a zero or Pascal NIL value in an entry. If zero NRB's are to be waited upon (polling) then the first entry in the table has a NIL.

The Data Mover is structured so that the wait logic allows NRB's for different connections to be waited upon concurrently.

TRANSPORT SERVICE INTERFACE

The Data Mover protocol is designed to function in environments where very long propagation delays, extremely high speeds, and high error rates are all concurrently encountered. Most conventional protocols do not function well in such an environment.

In order to resolve the long delay and high error rate problem, the Data Mover Transport uses a "continuous" type of protocol - one in which only blocks received in error are retransmitted, rather than "backing up" to the point where the erroneous block was sent and sending all subsequent blocks. The Data Mover Transport uses a dual numbering scheme for blocks. There is a "physical block number" which is incremented by one each time Transport issues an N-CONNECT or N-WRITE request. In addition, there is a "logical block number" which is incremented by one each time the Transport application issues a write type Transport service request.

The Transport layer has a main responsibility of providing guaranteed, correct data delivery to the peer process. To accomplish this, it provides:

1. The ability to negotiate transmission block sizes agreeable to both sides of the connection.

2. The ability to segment an arbitrarily large block of application data into messages suitable for transmission, and to reassemble the data block for the receiving application.

3. An acknowlegement scheme so that messages lost in transit can be retransmitted.

4. A flow control facility so that the receiving user or Data Mover is not overrun with data.

5. Alternate path retry capability, where alternate routes to a destination are tried if a primary one fails.

Transport will deliver data in correct order at the pace no greater than that desired by the receiver. It will either deliver the data successfully or it will report that communications with the remote transport have been lost over all possible paths. In that event, the status of data not explicitly acknowledged by the remote application is not known.

Transport service is distinct from the higher layers of networking software in that connections can only be made to other parties whose exact physical address on the network is known, and the party connected to must be present and soliciting input at the time. Higher layers of software allow more flexibility in the addressing of remote application programs.

Transport service uses the lower Network service to establish connections on the network and to frame data for delivery on the network.

A T-OFFER request allocates the resources so that an explicit path to the transport application through the lower network layers is established. At that time, the T-OFFER waits for a matching T-CONNECT request. T-CONNECT accepts an (optional) buffer of data to be sent to the OFFERing receiver, and a list of routes (addresses) that specify the network path or paths that may be used to contact the receiver. When the CONNECT data arrives at the receiver, it completes the OFFER and presents the connect data to the receiving application.

The receiving application responds with a T-CONFIRM request which indicates the intent to complete the connection. If the connection is not desired, the receiver may issue T-DISCONNECT instead. T-CONFIRM may have data sent along with it. The connecting application will receive the data with a T-READ request that completes with a Confirm Indication.

Now that the connection is open, data transfer may proceed in a full duplex manner, with flow control, segmenting, and retransmission taking place concurrently in both directions. So long as the connection remains in effect, a T-WRITE request by one of the parties will place the data in a transmission queue so that it will eventually satisfy a T-READ issued by the peer Data Mover.

When the connection is complete, then either side may begin connection termination by issuing a T-CLOSE request. T-CLOSE may have data associated with it. Following the first T-CLOSE, the issuing Packet Driver may not write any more data on that connection (except a T-DISCONNECT). The other side will receive the Close data with a Close Indication following the normal reception of all normally written data. It may write as much data to the originating connection as it chooses, but it will not be able to receive any more data from the originator. Connection termination completes when the second Packet Driver issues its own T-CLOSE request. The connection is terminated when the originating party receives the T-CLOSE request following the reception of all normal data preceding the CLOSE.

Transport also offers a T-DISCONNECT service for use in abruptly terminating a connection at any time. Either party may issue a T-DISCONNECT request. Data may be included with the T-DISCONNECT, but delivery of the data is not guaranteed. Disconnect is preemptive in that it will be presented to the receiver as soon as it arrives at the receiving side. Data controlled by transport, regardless of the direction in which it is sent, will be lost.

T-CONNECT - ESTABLISH TRANSPORT CONNECTION

The T-CONNECT request is used to initiate the establishment of a connection to another remote Transport

application that has issued a T-OFFER request. When invoked, Transport service will establish a Network connection to the remote host and send it messages to establish the connection. It will send information that will allow data to flow in an orderly, full duplex manner between applications.

The parameters of the T-CONNECT request are:

T-CONNECT	<NRBBUFA	- address of outgoing Pdata>
	<NRBLEN	- length of outgoing Pdata>
	<NRBDMODE	- datamode of Pdata>
	<NRBPROTA	- address of outgoing Odata>
	<NRBPROTL	- length of outgoing Odata>
	<NRBNREF	- N-ref desired for this connection>
	<NRBBLKO	- Maximum transmission size acceptable>
	<NRBBLKI	- Maximum reception size acceptable>
	<NRBMXRAT	- limit on transmission speed>
	<NRBCONN1	- Physical Address Map list of connection>
	<NRBCONN2	- remote N-ref to connect to>
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- set to zero
	NRBBLKO	- maximum transmission Pdata size
	NRBBLKI	- maximum reception Pdata size
	NRBMXRAT	- Max transmission speed of path

When the T-CONNECT is issued, the application specifies two buffer lengths and addresses - one for Pdata and one for Odata. The Odata is always sent and received in the "octet" (8 bit byte) format that is suitable for the particular host.

The maximum amount of Odata that may be sent is dependent on values generated during system generation on both the local and remote copies of the Data Mover. The Session and Transport layers require 256 bytes of Octet data less the Network layer Odata requirements.

NRBBUFA specifies the address of the Pdata that is to be sent to the offering party.

NRBLEN specifies the length of the Pdata (in addresable units) that is to be sent with the T-CONNECT request. The amount of Pdata that may be sent with a T-CONNECT request is limited; in general, the length of the data can be no greater than the maximum associated data size supported by the Network level connection.

NRBDMODE specifies the Datamode to be used on the Pdata during transfer. Datamod is discussed in the section on the Data Mover Network Request Block.

NRBPROTA and **NRBPROTL** specify the length and address of the Odata to be sent to the offering Transport application. Buffers of Odata must always be provided by the application. The size of Odata is limited to the maximum size of Data transmission that is supported by the Network connection in use.

NRBNREF contains the N-ref that is to be associated with the Network connection that is to be used by Transport service on this host. An explicit N-ref must be requested; if the one requested is already in use, then the connection request will be rejected. The remote application must supply the same N-ref in the NRBNREF field of its T-OFFER request.

NRBMXRAT is a value that may be specified by the user to limit the rate at which data is introduced to the network. Values in this field are in 1000's of bits per second. If this value is set to zero when the T-CONNECT is issued, then the maximum transmission rate possible with the active route will be used. When the T-CONNECT completes, NRBMXRAT will contain the transmission speed limit in 1000's of bits per second. As a result, sophisticated applications can recognize that communications is taking place through a lower speed link and adjust their transmission strategy accordingly.

If Transport service finds that another route must be used because of a communications failure on the first, then NRBMXRAT will be changed to reflect the speed limit on the new route. The Transport application will receive no other notification of the change of route.

NRBBLKO and **NRBBLKI** can be used to specify the maximum Pdata sizes that are acceptable to the application for output and input, respectively. They will be sent to the offering application, who may also provide values for these fields in the T-OFFER request. When the T-OFFER completes, or the connecting party has the T-READ complete which reads the Offeror 's T-CONFIRM, then these fields will contain the minimum of the two sizes specified by the two parties. Once these values are negotiated, the Data Mover will not permit a block of Pdata greater than this maximum to be transmitted on the network. The Data Mover also uses this upper limit on transmission size to improve connection and buffer management.

Two default values are available for these fields. If a zero is specified in either one of these fields, then an installation defined default will be used. If a -1 is contained in this field, then the maximum size allowable by this installation will be used. The -1 value is typically used by Offering applications that wish the connecting application to specify the maximum input and output block sizes.

NRBCONN1 contains a list of PAM's that represent possible routes to the same host. Transport will attempt to establish and use an N-connection for the first of the entries in this list. If loss of communication is detected during the connection or at any later time in the lifetime of thc connection, then Transport will establish another N-connection with the next PAM in the list and attempt to continue the connection. This switching of N-connections will take place without affecting the Transport application. If the entire list of PAM's is tried and none of the paths result in the re-establishment of communications,. then the T-connection will fail with a Disconnect Indication.

NRBCONN1 itself contains an address pointing to the start of a PAM list. The format of this PAM list is illustrated in FIGURE 6C; the format of a PAM list entry is illustrated in FIGURE 6D; the format of a PAM route is illustrated in FIGURE 6E. The entire PAM list is in octet format. The first two octets 720 contain the length of the PAM list, including the length field. The third octet 721 contains the number of individual PAM's in the PAM list. Following that are the individual PAM's, 722 packed together so that adding an individual PAM length to its start will generate the address of the next individual PAM. Refer to the Network Service chapter for more information on the format of individual PAM's.

NRBCONN2 contains the N-ref that the remote, offering Transport application has requested in its NRBNREF when the T-OFFER was issued. The application must know this value either by prearrangement or through the use of Session services.

T-OFFER - SOLICIT INCOMING TRANSPORT CONNECTION

The T-OFFER request is used by a process that wants to establish a "passive" connection - to wait for another, "active" party to request that the connection be completed. It is a read-type request in the sense that it remains outstanding until an incoming T-connect indication arrives, and that data sent by the remote T-CONNECT will be used to fill buffers specified by the T-OFFER.

The parameters in the NRB that are used in T-OFFER are as follows:

T-OFFER	<NRBBUFA	- address for incoming Pdata>
	<NRBBUFL	- length of buffer to hold Pdata>
	(NRBPROTA	- address for incoming Odata>
	<NRBPROTL	- length of buffer to hold Odata>
	<NRBTIME	- number of seconds offer outstanding>
	<NRBNREF	- local N-ref desired for connection>
	<NRBBLKO	- Maximum transmission size acceptable>
	<NRBBLKI	- Maximum reception size acceptable>
	<NRBMXRAT	- limit on transmission speed>
	<NRBCONN1	- buffer to hold route PAM>
	<NRBCONN2	- N-ref of remote (connecting) party>
	Options: Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- contains Connect Indication
	NRBLEN	- length of incoming Pdata
	NRBPROTL	- length of Odata received
	NRBNREF	- N-ref assigned this connection
	NRBBLKO	- maximum transmission Pdata size
	NRBBLKI	- maximum reception Pdata size
	NRBMXRAT	- Max transmission speed of path
	NRBCONN1	- First PAM used to connect.
	NRBCONN2	- Connector's N-ref

NRBBUFA contains the address of the buffer to be used to store the Pdata with the incoming T—connect.

NRBBUFL contains the length of the buffer provided to receive the incoming Pdata. If the data that arrives exceeds this length, then the buffer will be filled up to NRBBUFL, NRBLEN will contain the amount of data that could have been received, and NRBSTAT will contain an error indicating that Pdata was truncated.

NRBPROTA and **NRBPROTL** contain the address and length of a buffer to hold the incoming Odata. As with Pdata, only NRBPROTL octets of data will be placed in the buffer if a greater amount of information arrives. When the T-OFFER completes successfully, NRBPROTL will contain the number of octets of Odata actually received.

NRBTIME contains the time interval that the T-OFFER is to remain outstanding. If this period of time elapses and no successful T-connection is made to satisfy, then the T-OFFER will terminate with an error code in NRBSTAT. The value in NRBTIME is measured in seconds.

NRBNREF contains the N-ref that is to be used for this transport connection. The application issuing the T-CONNECT will have to supply a matching N-ref in its NRBNREF. If the N-ref requested is already in use, then the request will be aborted with an error in NRBSTAT.

NRBBLKO and **NRBBLKI** contain the maximum size block that the Transport application wishes to send with a single write or read-type request, respectively. When the T-OFFER completes with the connect information, these fields will contain the minimum of the two values specified by the offering and connecting applications. Attempts to transmit more data than the maximum in NRBBLKO will be rejected as an error. The size of the blocks sent and received is only limited by the maximum address size on the host machine.

Two default values are available for these fields. If a zero is specified in either one of these fields, then an

installation defined default will be used. If a -1 is contained in this field, then the maximum size allowable by this installation will be used. The -1 value is typically used by Offering applications that wish the connecting application to specify the maximum input and output block sizes.

NRBMXRAT contains the fastest data rate that may be used for transmission from the offering program. If this field is set to zero when the T-OFFER is issued, then it will contain the maximum rate that can be supported by the communications path in use when the connection is established. If a nonzero value is provided, then completion will result in the minimum of the user specified and communications link speeds. This rate specified only applies to transmission; the connecting end of the connection has its own NRBMXRAT that may be used to limit transmission speed in the other direction.

NRBCONN1 specifies the address of a buffer that will hold the incoming PAM. This PAM will be .the one that first established successful communications between the two Transport facilities. It may be used by the offering application to determine if the originator is permitted to proceed with the connection. Space for 64 octets should be provided to hold a PAM of maximum length.

NRBCONN2 contains the N-ref used by the remote host whose T-CONNECT will cause this offer to complete. If a Connect Indication specifying the proper local N-ref arrives which does not contain the proper remote N-ref, then the incoming message will be ignored. If a zero is specified in NRBCONN2, then any remote N-ref will be acceptable. In that case, NRBCONN2 will contain the remote N-ref when the T-OFFER completes.

T-CONFIRM - COMPLETE TRANSPORT CONNECTION.

T-CONFIRM is issued by the offering application in response to a successfully completed T-OFFER. It signals the intent of the offering application to go ahead with the connection, and to have data sent back to the originating application. Either T-CONFIRM or T-DISCONNECT must be issued in response to a successful T-OFFER; any other request will be rejected. A T-CONFIRM is not valid at any time other than immediately after the T-OFFER.

T-CONFIRM is a write type command, and both Odata and Pdata may be sent back to the originating application.

The parameters for T—CONFIRM are:

T—CONFIRM	(NRBBUFA	- address of outgoing Pdata>
	<NRBLEN	- length of outgoing Pdata>
	<NRBDMODE	- datamode of Pdata>
	<NRBPROTA	- address of outgoing Odata>
	<NRBPROTL	- length of outgoing Odata>
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	-success/failure code
	NRBIND	- set to zero
	NRBBLKO	- max transmission size for connection
	NRBBLKI	- max received size for connection

NRBBUFA contains the address of Pdata to be returned to the originating application.

NRBLEN contains the length of the transmitted Pdata in addressable units. The length of data must be less than or equal to NRBBLKO. The theoretical limit on Pdata size is the addressability range of the user program.

NRBDMODE contains the Datamode to be used for the transmission of the Confirm Pdata.

NRBPROTA and **NRBPROTL** contain the address and number of octets of Odata to be sent to the originating application.

NRBBLKO is ignored when the T-CONFIRM request is issued.

T-READ - SOLICIT INDICATION FOR T-CONNECTION

The T-READ is a request that informs the Data Mover that the Transport application is ready to process additional input from a Transport connection. In general, such input will be a Data Indication of some type, indicating that data has arrived from the remote application. However, it can also be a Disconnect Indication indicating that the connection has been lost for some reason.

One of the purposes of the T-READ is to allow the application to accept incoming data at a constrained rate; however, it is essential that a T-READ be issued to service incoming data on a timely basis. The Data Mover Transport service places a time interval limit (referred to as READTO) which causes a connection to be terminated if no T-READ is issued to receive an incoming indication within READTO seconds of elapsed time.

In addition to Odata and Pdata, the T-READ request will complete with an "indication" in NRBIND, which indicates the type of Transport request that was issued to send this data. A Confirm, Data, Close, or Disconnect indication may be received along with its associated Odata and Pdata.

The format of the T-READ request is:

T-READ	<NRBBUFA	- address for incoming Pdata>
	<NRBBUFL	- length of buffer to hold Pdata>
	<NRBPROTA	- address for incoming Odata>
	<NRBPROTL	- length of buffer to hold Odata>
	<NRBTIME	- number of seconds offer outstanding>
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- contains Connect Indication
	NRBBUFA	- address of Pdata (if ptr mode selected)
	NRBLEN	- length of incoming Pdata
	NRBPROTL	- length of Odata received

NRBBUFA contains the address of the buffer that will hold the incoming Pdata.

NRBBUFL is used to specify the maximum length of the user supplied buffer to hold Pdata.

NRBPROTA and **NRBPROTL** are used to specify a buffer for incoming Odata. If more than NRBPROTL octets of data arrive, the Odata will be truncated, NRBSTAT will contain an error, and NRBPROTL will contain the length that could have been received. NRBTIME specifies the maximum length of elapsed time that the T-READ is to remain outstanding. If this interval passes and there is no

indication to give to the Transport application, then the T-READ completes with an error code in NRBSTAT. The connection is not terminated; subsequent T-READ's and T-WRITE's may be issued at any time. If zero is present in this field, the read will remain outstanding indefinitely. The value of this field is in seconds.

T-WRITE - SEND DATA TO REMOTE APPLICATION

The T-WRITE request is used when a connection is fully established to send Odata and Pdata to the remote application. When issued, the T-WRITE completes as soon as transport has accepted responsibility for the data. The data will be delivered correctly and in order so long as the connection remains intact.

The parameters used by T-WRITE are:

T-WRITE	<NRBBUFA	- address of outgoing Pdata>
	<NRBLEN	- length of outgoing Pdata>
	<NRBDMODE	- datamode of Pdata>
	<NRBPROTA	- address of outgoing Odata>
	<NRBPROTL	- length of outgoing Odata>
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- set to zero
	NRBBUFA	- next buffer addr (if ptr mode selected)

NRBBUFA contains the address of Pdata to be returned to the originating application.

NRBLEN contains the length of the transmitted Pdata in addressable units. The length of data must be less than or equal to NRBBLKO. The theoretical limit on Pdata size is the addressability range of the user program.

NRBDMODE contains the Datamode to be. used for the transmission of the Pdata.

NRBPROTA and **NRBPROTL** contain the address and number of octets of Odata to be sent to the remote application.

T-CLOSE - GRACEFULLY TERMINATE TRANSPORT CONNECTION

When either party of a connection determines that it has only one block or no more data to send, it may issue a T-CLOSE request. The T-CLOSE is a write type request and may have data associated with it, following the same rules as WRITE requests. After the CLOSE is issued, the issuer may not issue any more WRITE type requests except DISCONNECT. The issuer may continue to read data indefinitely.

All pending written data is provided to the second application, in order of transmission. The last data item received will have a Close Indication associated with it. No further reads will be accepted for this connection. The second application may issue any number of WRITEs to the originating application. When the second application has no more data to send, it issues a CLOSE (with optional data.) At that time, the connection is terminated from the vantage of the second application .

The originating application will receive all data written by the second application in proper order. The last data received will contain a Close Indication. When the Close Indication is read, the connection is

terminated from the viewpoint of the originating application .

The parameters used by T-CLOSE are:

T—CLOSE	<NRBBUFA	- address of outgoing Pdata>
	<NRBLEN	- length of outgoing Pdata>
	<NRBUBIT	- Pdata unused bit count>
	<NRBDMODE	- datamode of Pdata>
	<NRBPROTA	- address of outgoing Odata>
	<NRBPROTL	- length of outgoing Odata>
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- set to zero
	NRBBUFA	- contains zero (if ptr mode selected)

NRBBUFA contains the address of Pdata to be returned to the . originating application.

NRBLEN contains the length of the transmitted Pdata in addressable units. The length of data must be less than or equal to NRBBLKO. The theoretical limit on Pdata size is the addressability range of the user program.

NRBDMODE contains the Datamode to be used for the transmission of the Close Pdata.

NRBPROTA and **NRBPROTL** contain the address and number of octets of Odata to be sent to the remote application.

T-DISCONNECT - ABORT TRANSPORT CONNECTION

When either party of the connection determines that the connection must be abruptly terminated, it may issue the T-DISCONNECT request to do so. T-DISCONNECT is a preemptive service in that all reads, writes, and data buffers in progress" on the connection as the

Disconnect request is discovered are discarded.

If an application issues a T-DISCONNECT, then the T-READ request (if any) that is outstanding request will terminate with an error in NRBSTAT. Any further attempts to issue requests against the connection will be rejected.

The remote application will continue to receive any data in progress until the remote application's Transport service detects the incomming Disconnect Indication. Any connection indications waiting to be delivered to the user will be discarded; the next T-READ will be complete with a Disconnect Indication, a normal status in NRBSTAT, and the Odata and Pdata supplied by the originator of the T-DISCONNECT.

Disconnect service is different from other requests in that delivery of the Disconnect Odata and Pdata is NOT guaranteed. Such data may be lost in such a way that the local Transport service cannot be sure if it was delivered or not.

The T-DISCONNECT is valid at any time in the lifetime of the connection that a Write-type request (Confirm, Write, Close) may be issued. It is also valid following a T-CLOSE request. In all cases data may not be sent to the remote application but the connection will be terminated.

T-DISCONNECT has two principal uses in practice; first it may be sent instead of a Confirm if the application does not wish to accept the connection. Secondly, it may be used when an application detects "garbage" being sent to it and it is not receiving a meaningful response from the remote application.

The parameters used by T-DISCONNECT are:

```
T-DISCONNECT   <NRBBUFA      - address of outgoing Pdata>
               <NRBLEN       - length of outgoing Pdata>
               <NRBDMODE     - datamode of Pdata>
               <NRBPROTA     - address of outgoing Odata>
               <NRBPROTL     - length of outgoing Odata>
               Options:
                  Wait | NoWait

 Results:       NRBSTAT      - success/failure code
                NRBIND       - set to zero
```

NRBBUFA contains the address of Pdata to be returned to the originating application.

NRBLEN contains the length of the transmitted Pdata in addressable units. The length of Pdata sent is limited to the minimum of the internal Data Mover buffer size and the maximum Associated Data size supported by the Network connection in use between the two applications.

NRBDMODE contains the Datamode to be used for the transmission of the Disconnect Pdata.

NRBPROTA and **NRBPROTL** contain the address and number of octets of Odata to be sent to the remote application.

SESSION SERVICE INTERFACE

Session service is fairly similar to the services offered by the Transport Layer. The major difference is the flexibility and convenience with which remote applications may be addressed. In addition to Transport service, the Session Layer provides:

1. The ability to address a remote application by two character string names: a "host" name and an "application" name. Network addressing and alternate path considerations are handled by the Session Layer.

2. The ability to construct "server" applications by OFFERing the same service many times.

3. The ability to conserve network resources until a connection is actually established.

An application wishing to respond to a connection user issues an S-OFFER request, which "advertises" the application name supplied by the application as being available on that "host". The same application name may be advertised many times, where connections will be completed in the order the OFFERs are issued.

A different application that wishes to connect to this application issues a S-CONNECT request, providing the "host" and "application" names. Data may be provided by the S-CONNECT, which will fill the buffers supplied by the S-OFFER request.

After the Connect data has arrived, the service requests are identical to Transport: the receiver issues an S-CONFIRM which satisfies the connector's S-READ. Both sides may concurrently S-READ and S-WRITE data. The connection may be terminated with a pair of S-CLOSE requests or a single S-DISCONNECT request.

S-CONNECT - ESTABLISH SESSION CONNECTION

The S-CONNECT request is used to initiate the establishment of a connection to another remote Session application that has issued a S-OFFER request. When invoked, Session service will establish a Transport connection to the remote host and send it messages to establish the connection. 'It will send information that will allow data to flow in an orderly, full duplex manner between applications.

Data Mover connection philosophy works with an "active" and a "passive" party of any connection. The "passive" party issues an S-OFFER request and waits for a subsequent connection to arrive. Any other node with access to the host with the offering application may then connect to it. The "active" party then issues a S-CONNECT which causes the connection to be opened. If an S-CONNECT is issued to connect to an application that is not yet present, then the connection will fail; thus the S-OFFER must precede the S-CONNECT chronologically.

The parameters of the S-CONNECT request are:

S-CONNECT	<NRBBUFA	- address of outgoing Pdata>
	<NRBLEN	-length of outgoing Pdata>
	<NRBDMODE	- datamode of Pdata>
	<NRBPROTA	- address of outgoing Odata>
	<NRBPROTL	- length of outgoing Odata>
	<NRBBLKO	- Maximum transmission size acceptable>
	<NRBBLKI	- Maximum reception size acceptable>
	<NRBMXRAT	- limit on transmission speed>
	<NRBCONN1	- alphanumeric "host" name
	<NRBCONN2	- alphanumeric "application" name
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- set to zero
	NRBNREF	- S-ref (Session ID) assigned
	NRBBLKO	- maximum transmission Pdata size
	NRBBLKI	- maximum reception Pdata size
	NRBMXRAT	- Max transmission speed of path.

When the S-CONNECT is issued, the application specifies two buffer lengths and addresses - one for Pdata and one for Odata. The Odata is always sent and received in the "octet" (8 bit byte) format that is suitable for the particular host. The amount of Odata that may be sent over any connection is dependent on installation parameters specified for both the local and remote copies of the Data Mover.

NRBBUFA specifies the address of the Pdata that is to be sent to the offering party.

NRBLEN specifies the length of the Pdata (in addressable units) that is to be sent with the S-CONNECT request. The amount of Pdata that may be sent with a S-CONNECT request is limited; in general, the length of the data can be no greater than the maximum associated data size supported by the Network level

connection.

NRBDMODE specifies the Datamode to be used on the Pdata during transfer. Datamode is discussed in the section on the Data Mover Request Block .

NRBPROTA and **NRBPROTL** specify the length and address of the Odata to be sent to the offering Session application. Buffers of Odata must always be provided by the application. The maximum amount of Odata that may be sent over a Session connection is limited by installation parameters on both the local and remote copies of the Data Mover.

NRBMXRAT is a value that may be specified by the user to limit the rate at which data is introduced to the network. Values in this field are in 1000's of bits per second. If this value is set to zero when the S-CONNECT is issued, then the maximum transmission rate possible with the active route will be used. When the S-CONNECT completes, NRBMXRAT will contain the transmission speed limit in 1000's of bits per second. As a result, sophisticated applications can recognize that communications is taking place through a lower speed link and adjust their transmission strategy accordingly.

If Session service finds that another route must be used because of a communications failure on the first, then NRBMXRAT will be changed to reflect the speed limit on the new route. The Session application will receive no other notification of the change of route.

NRBBLKO and **NRBBLKI** can be used to specify the maximum Pdata sizes that are acceptable to the application for output and input, respectively. They will be sent to the offering application, who may also provide values for these fields in the S-OFFER request. When the S-OFFER completes, or the connecting party has the S-READ complete which reads offeror 's S-CONFIRM, then these fields will contain the minimum of the two sizes specified by the two parties. Once these values are negotiated, the Data Mover will not permit a block of Pdata greater than this maximum to be transmitted on the network. The Data Mover also uses this upper limit on transmission size to improve connection and buffer management .

Two default values are available for these fields. If a zero is specified in either one of these fields, then an installation defined default will be used. If a -1 is contained in this field, then the maximum size allowable by this installation will be used. The -1 value is typically used by Offering applications that wish the connecting application to specify the maximum input and output block sizes.

NRBCONN1 specifies an eight character alphanumeric name of the host that may be running the matching S-OFFERed application. This name will be an element of a Network Configuration provided when the Data Mover was started or generated. Session service will determine if the application name provided in NRBCONN2 is an available, S-OFFERed, application.

NRBCONN2 contains the name of the application to which the connection will be made. The offering application will have provided this name in NRBCONN2 of its S-OFFER request. No restrictions are made on the application name offered. If the same application name is offered several times on the host or host group connected to, then the oldest' S-OFFER request will accept the S—connection.

S-OFFER - SOLICIT INCOMING SESSION CONNECTION

The S-OFFER request is used by a process that wants to establish a "passive" connection - to wait for another, "active" party to request that the connection be completed. It is a read-type request in the sense that it remains outstanding until an incoming S-connect indication arrives, and that data sent by the remote S-CONNECT will be used to fill buffers specified by the S-OFFER.

The parameters in the NRB that are used in S-OFFER are:

S-OFFER	<NRBBUFA	- address for incoming Pdata>
	<NRBBUFL	- length of. buffer to hold Pdata>
	<NRBPROTA	- address for incoming Odata> .
	<NRBPROTL	- length of buffer to hold Odata>
	<NRBTIME	- number of seconds offer outstanding>
	<NRBBLKO	- Maximum transmission size acceptable>
	<NRBBLKI	- Maximum reception size acceptable>
	<NRBMXRAT	- limit on transmission speed>
	<NRBCONN2	- Application name to offer>
	Options: Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- contains Connect Indication
	NRBLEN	- length of incoming Pdata
	NRBPROTL	- length of Odata received
	NRBNREF	- S-ref assigned this connection
	NRBBLKO	- maximum transmission Pdata size
	NRBBLKI	- maximum reception Pdata size
	NRBMXRAT	- Max transmission speed of path
	NRBCONN1	- Name of host where S-conn originated
	NRBCONN2	- User-id of program originating S-conn

NRBBUFA contains the address of the buffer to be used to store the Pdata with the incoming S-connect.

NRBBUFL contains the length of the buffer provided to receive the incoming Pdata. If the data that arrives exceeds this length, then the buffer will be filled up to NRBBUFL, NRBLEN will contain the amount of data that could have been received, and NRBSTAT will contain an error indicating that Pdata was truncated.

NRBPROTA and **NRBPROTL** contain the address and length of a buffer to hold the incoming Odata. As with Pdata, only NRBPROTL octets of data will be placed in the buffer if a greater amount of information arrives. When the S-OFFER completes successfully, NRBPROTL will contain the number of octets of Odata actually received.

NRBTIME contains the time interval that the S-OFFER is to remain outstanding. If this period of time elapses and no successful S-connection is made to satisfy, then the S-OFFER will terminate with an error code in NRBSTAT. The value in NRBTIME is measured in seconds .

NRBNREF contains the N-ref that is to be used for this session connection.

NRBBLKO and **NRBBLKI** contain the maximum size block that the Session application wishes to send with a single write or read-type request, respectively. When the S-OFFER completes with the connect information, these fields will contain the minimum of the two values specified by the offering and connecting applications. Attempts to transmit more data than the maximum in NRBBLKO will be rejected as an error.

Two default values are available for these fields. If a zero is specified in either one of these fields, then an installation defined default will be used. If a -1 is contained in this field, then the maximum size allowable by this installation will be used. The -1 value is typically used by Offering applications that wish the connecting application to specify the maximum input and output block sizes.

NRBMXRAT contains the fastest data rate that may be used for transmission from the offering program. If

this field is set to zero when the S-OFFER is issued, then it will contain the maximum rate that can be supported by the communications path in use when the connection is established. If a nonzero value is provided, then completion will result in the minimum of the user specified and communications link speeds. This rate specified only applies to transmission; the connecting end of the connection has its own NRBMXRAT that may be used to limit transmission speed in the other direction.

NRBCONN1 is ignored when the S-OFFER request is made. If the S-OFFER request completes successfully, it will have the eight character alphanumeric name of the host from which the S-CONNECT originated. Session applications may examine this field to determine if the connection should be accepted or not.

NRBCONN2 contains the name of the application which is to be offered. The connecting application program must supply a matching name in NRBCONN2 of its S-CONNECT request for the connection to succeed. No restrictions are placed on the name provided. If the same application name is offered several times, then the oldest matching OFFER request on

S-CONFIRM - COMPLETE SESSION CONNECTION

S-CONFIRM is issued by the offering application in response to a successfully completed S-OFFER. It signals the intent of the offering application to go ahead with the connection, and to have data sent back to the originating application. Either S-CONFIRM or S-DISCONNECT must be issued in response to a successful S-OFFER; any other request will be rejected. An S-CONFIRM is not valid at any time other than immediately after the S-OFFER.

S-CONFIRM is a write type command, and both Odata and Pdata may be sent back to the originating application.

The parameters for S-CONFIRM are:

```
S-CONFIRM     <NRBBUFA    - address of outgoing Pdata (move mode)
              <NRBLEN     - length of outgoing Pdata>
              <NRBDMODE   - datamode of Pdata>
              <NRBPROTA   - address of outgoing Odata>
              <NRBPROTL   - length of outgoing Odata>
              Options:
                Wait | NoWait

Results:      NRBSTAT     - success/failure code
              NRBIND      - set to zero
```

NRBBUFA contains the address of Pdata to be sent to the receiving application.

NRBLEN contains the length of the transmitted Pdata in addressable units. The length of data must be less than or equal to NRBBLKO. If move mode is specified in the NRBREQ field of the S-OFFER, then the theoretical limit on Pdata size the addressability range of the user program.

NRBDMODE contains the Datamode to be used for the transmission of the Confirm Pdata.

NRBPROTA and **NRBPROTL** contain the address and number of octets of Odata to be sent to the

originating application.

S-READ - SOLICIT INDICATION FOR S-CONNECTION

The S-READ is a request that informs the Data Mover that the Session application is ready to process additional input from a Session connection. In general, such input will be a Data Indication of some type, indicating that data has arrived from the remote application. However, it can also be a Disconnect Indication indicating that the connection has been lost for some reason.

One of the purposes of the S-READ is to allow the application to accept incoming data at a constrained rate; however, it is essential that an S-READ be issued to service incoming data on a timely basis. The Data Mover Transport service places a time interval limit (referred to as READTO) which causes a connection to be terminated if no S-READ is issued to receive an incoming indication within READTO seconds of elapsed time.

In addition to Odata and Pdata, the S-READ request will complete with an "indication" in NRBIND, which indicates the type of Session request that was issued to send this data. A Confirm, Data, Close, or Disconnect indication may be received along with its associated Odata and Pdata.

The format of the S-READ request is:

S-READ	<NRBBUFA	- address for incoming Pdata (move mode)>
	<NRBBUFL	- length of buffer to hold Pdata>
	<NRBNREF>	- Sref (Session-id) assigned to session
	<NRBPROTA	- address for incoming Odata>
	<NRBPROTL	- length of buffer to hold Odata>
	<NRBTIME	- number of seconds offer outstanding>
	Options:	
	Wait | NoWait	
Results:	NRBSTAT	- success/failure code
	NRBIND	- contains Connect Indication
	NRBLEN	- length of incoming Pdata
	NRBPROTL	- length of Odata received

NRBBUFA contains the address of the buffer that will hold the incoming Pdata.

NRBBUFL is used specify the maximum length of the user supplied buffer to hold Pdata. If Pointer mode is in effect, this field is ignored.

NRBPROTA and **NRBPROTL** are used to specify a buffer for incoming Odata. If more than NRBPROTL octets of data arrive, the Odata will be truncated, NRBSTAT will contain an error, and NRBPROTL will contain the length that was received. NRBTIME specifies the maximum length of elapsed time that the S-READ is to remain outstanding. If this interval passes and there is no indication to give to the Session application, then the S-READ completes with an error code in NRBSTAT. The connection is not terminated; subsequent S-READ's and S-WRITE's may be issued at any time. If zero is present in this field, the read will remain outstanding indefinitely. The value of this field is in seconds.

S-WRITE - SEND DATA TO REMOTE APPLICATION

The S-WRITE request is used when a connection is fully established to send Odata and Pdata to the remote application. When issued, the S-WRITE completes as soon as session has accepted responsibility for the data. The data will be delivered correctly and in order so long as the connection remains intact.

The parameters used by S-WRITE are:

```
S-WRITE     <NRBBUFA     - address of outgoing Pdata>
            <NRBLEN      - length of outgoing Pdata>
            <NRBDMODE    - datamode of Pdata>
            <NRBNREF>    - Sref (Session-id) assigned to session
            <NRBPROTA    - address of outgoing Odata>
            <NRBPROTL    - length of outgoing Odata>
            Options:
              Wait | NoWait

Results:    NRBSTAT      - success/failure code
            NRBIND       - set to zero
```

NRBBUFA contains the buffer address where Pdata to be returned to the originating application.

NRBLEN contains the length of the transmitted Pdata in addressable units. The length of data must be less than or equal to NRBBLKO. The theoretical limit on Pdata size the addressability range of the user program.

NRBDMODE contains the Datamode to be used for the transmission of the Pdata.

NRBPROTA and **NRBPROTL** contain the address and number of octets of Odata to be sent to the remote application.

S-CLOSE - GRACEFULLY TERMINATE SESSION CONNECTION

When either party of a connection determines that it has only one block or no more data to send, it may issue S-CLOSE request. The S-CLOSE is a write type request and may have data associated with it, following the same rules as WRITE requests. After the CLOSE is issued, the issuer may not issue any more WRITE type requests except DISCONNECT. The issuer may continue to read data indefinitely.

All pending written data is provided to the second application, in order of transmission. The last data item received will have a Close Indication associated with it. No further reads will be accepted for this connection. The second application may issue any number of WRITEs to the originating application. When the second application has no more data to send, it issues a CLOSE (with optional data.) At that time, the connection is terminated from the vantage of the second application .

The originating application will receive all data written by the second application in proper order. The last data received will contain a Close Indication. When the Close Indication is read, the connection is terminated from the viewpoint of the originating application

The parameters used by S-CLOSE are:

```
SCLOSE      <NRBBUFA    - address of outgoing Pdata>
            <NRBLEN     - length of outgoing Pdata>
```

```
                <NRBDMODE   - datamode of Pdata>
                <NRBNREF>   - Sref (Session-id) assigned tosession
                <NRBPROTA   - address of outgoing Odata>
                <NRBPROTL   - length of outgoing Odata>
                Options:
                  Wait | NoWait

Results:        NRBSTAT     - success/failure code
                NRBIND      - set to zero
                NRBBUFA     - contains zero (if ptr mode selected)
```

NRBBUFA contains the address of the last buffer of Pdata to be sent to . the remote application. .

NRBLEN contains the length of the transmitted Pdata in addressable units. The length of data must be less than or equal to NRBBLKO. The theoretical limit on Pdata size the addressability range of the user program.

NRBDMODE contains the Datamode to be used for the transmission of the Close Pdata.

NRBPROTA and **NRBPROTL** contain the address and number of octets of Odata to be sent to the remote application.

S-DISCONNECT - ABORT SESSION CONNECTION

When either party of the connection determines that the connection must be abruptly terminated, it may issue the S-DISCONNECT request to do so. S-DISCONNECT is a preemptive service in that all reads, writes, and data buffers "in progress" on the connection as the Disconnect request is discovered.

If an application issues a S-DISCONNECT, then the S-READ request (if any) that is outstanding request will terminate with an error in NRBSTAT. Any further attempts to issue requests against the connection will be rejected.

The remote application will continue to receive any data in progre s until the remote application's Session service detects the incom ng Disconnect Indication. Any connection indications waiting to be delivered to the user will be discarded; the next S-READ will be complete with a Disconnect Indication, a normal status in NRBSTAT, and the Odata and Pdata supplied by the originator of the S-DISCONNECT.

Disconnect service is different from other requests in that delivery of the Disconnect Odata and Pdata is NOT guaranteed. Such data may be lost in such a way that the local Session service cannot be sure if it was delivered or not.

The S-DISCONNECT is valid at any time in the lifetime of the connection that a Write-type request (Confirm, Write, Close) may be issued. It is also valid following a S-CLOSE request; in that case data may not be sent to the remote application but the connection will be terminated.

S-DISCONNECT has two principal uses in practice; first it may be sent instead of a Confirm if the application does not wish to accept the connection. Secondly, it may be used when an application detects "garbage" being sent to it and it is not receiving a meaningful response from the remote application.

The parameters used by S-DISCONNECT are:

```
S-DISCONNECT   <NRBBUFA    - address of outgoing Pdata>
```

```
                    <NRBLEN      - length of outgoing Pdata>
                    <NRBDMODE    - datamode of Pdata>
                    <NRBNREF>    - Sref (Session-id) assigned to session
                    <NRBPROTA    - address of outgoing Odata>
                    <NRBPROTL    - length of outgoing Odata>
                    Options:
                      Wait | NoWait

Results:            NRBSTAT      - success/failure code
                    NRBIND       - set to zero
```

NRBBUFA contains the address of Pdata to be sent to the remote application. Delivery of data is NOT guaranteed with Disconnect service.

NRBLEN contains the length of the transmitted Pdata in addressable units. The length of Pdata sent is limited to the minimum of the internal Data Mover buffer size and the maximum Associated Data size supported by the Network connection in use between the two applications.

NRBDMODE contains the Datamode to be used for the transmission of the Disconnect Pdata.

NRBPROTA and **NRBPROTL** contain the address and number of octets of Odata to be sent to the remote application, contain the address and number of octets of Odata to be sent to the remote application.

Application Layer (Packet Driver)

The Application Layer services are provided by the Packet Driver component, which is the Transport Protocol Optimizer's application layer. After the TPO communication link is established, the data from intercepted IP packets, along with routing information, is aggregated into a TPO buffer. When either the buffer is full, or TPO Session Services are available, the Packet Driver sends this buffer over the TPO connection to the peer Packet Driver, by using the S-WRITE session service and specifying the unique N-ref network reference number assigned when the TPO connection was established. This data will complete an S-READ session request previously issued by the peer Packet Driver.

This buffer can contain data from intercepted IP packets having source and destination IP addresses coming from and going to a number of different servers. It is up to the receiving Packet Driver to process the routing information received with each packet, and deliver the IP packets to the correct destination.

Protocol Message Descriptions

NETWORK LEVEL PROTOCOL

The present invention contains a method of forming Transport Protocol Optimizer messages that include Network Protocol data for the purpose of communicating with the peer-level Network layer on the remote side of the Transport Protocol Optimizer network.

The function of the network level protocols is to package a read or write type request containing Pdata and Odata into one or more network messages. In addition, it issues the messages required to establish a point to point connection between two TPO units.

NETWORK PROTOCOL FORMAT

Network protocols are located immediately following the Driver addressing information. The F2 message

ID is the indication to the Network layer of the start of Network Data. Network level protocol consists of a fixed field, followed by an optional Connect field that is sent when connections are first made.

All of these fields are present in every Network level network message. In the case were Network data is segmented into two network messages, ALL fields (except N-flags) are identical in both network messages to permit a higher degree of consistency checking.

FIGURE 6A illustrates the format of the Network message header. It consists of the following fields:

N-length: 700 contains the total number of octets of Network protocol in this particular network message. If N-connect data is included (discussed below) this field will not include the length of the N-connect data; that data will be part of the Odata. If the Odata is to be found in the message, then the first byte of Odata can be found by adding N-length to the start of the Network protocol.

N-level: 701 contains the version number of the Network protocol.

N-Plength: 702 contains the number of bits of useful information to be found in the Pdata. When the Network application presents a write type (N-CONNECT or N-WRITE) request to Network service, it supplies a length, unused bit count, and datamode in the NRB. Network service then calculates the number of bits of useful information that will be placed on the network after outgoing Assembly/Disassembly and Code Conversion have taken place. On input, it is the function of either the Network or Device Driver code to perform incoming Assembly/ Disassembly and Code Conversion and update the resulting number of bits in N-Plength. Once this processing is done, the Network service on the receiving side then places the Pdata length (in machine addressable units) and the unused bit count in the N-READ NRB.

If no Pdata is associated with the Network request, then N-Plength should be set to zeroes. If Pdata is present, the 24 bit field allows up to 16M-1 bits or 2,097,151 8-bit bytes to be sent with a single request.

N-datamode: 703 Datamode is used to define the Code Conversion requirements. Its use is documented in detail in the "NRB" section of the "User Interface" chapter. It is the responsibility of both the Network layer and the Device Driver to examine this field and modify it if changes are made to the data during transmission. It is also the responsibility of the Network layer software to perform software code conversion on incoming data before it is delivered to the Network application. When such software conversion is performed, the Network service should update the Datamode field showing that the conversion is performed before placing the final Datamode field in the user's NRB.

N-Olength: 704 contains the number of octets of Odata that are associated with this Network request. If N-connect data is present, it will be incorporated into the Odata and the 0-length will reflect the additional Odata. This field does NOT include the length of the N-protocol data that is discussed in this section.

This field is treated as an unsigned 16 bit binary number, allowing up to 65,535 (64K-1) octets of Odata to be sent with a single Network request. If no Odata is associated with the Network request (an unlikely circumstance) then this field will be set to zero.

N-TNref and N-RNref: 705 and **706** are used to identify a particular N-connection for incoming data. It uses the "socket" concept found in many other protocols, in that both transmitter and receiver contribute a field to the 32 bit "socket" that must match the receiver's version of the entire field before it is accepted as a connection.

When the originating Network application issues an N-CONNECT, it must provide (in NRBCONN2) the 16 bit value that is the receiver's N-ref, as well as its own transmitter's Nref in NRBNREF. In that connect message and all subsequent messages associated with that connection, Network service fills N-TNref and N-RNref with the values in NRBNREF and NRBCONN2 respectively.

When a message is returned back in the opposite direction on the same connection, then the values in these two fields will be reversed as the roles of transmitter and receiver have changed.

The receiving application also supplies a receiver's N-ref in the NRBNREF field of the N-OFFER, as well as the transmitter's N-ref in NRBCONN2. If an incoming network message does not match the 32 bit "socket" of any outstanding network connection, then the message is discarded.

This comparison to match incoming messages with network connections is done strictly on the basis of the 32 bit composite N-ref. Incoming messages from the driver, regardless of their point of origin, route, or incoming hardware interface, are considered to be for the same connection if the N-ref's match. Uniqueness of the N-ref on both machines is ensured as each copy of the Data Mover will only permit one connection to exist with a given local N-ref.

N-Ochecksum: 707 The Checksum field is used as a compromise between identifying loss of data integrity due to undetected hardware or network errors, and the conflict between thoroughly checksumming data and multi-megabit I/O performance. Added to this is the fact that it is not possible to meaningfully checksum the Pdata because of the Assembly/Disassembly and Code Conversion that may be performed upon it in transit. The N-Ochecksum field is generated by adding the 8 bit binary values found in each byte of the entire 64 octets of the Message Proper, excluding the value of the N-Ochecksum field itself, and storing the result modulo 256.

If the checksum field of an incoming network message does not match, then the entire message is discarded.

N-sequence: 708 is used to assure that composite (Case IV) messages are correctly reassembled upon reception by the receiver. When each side of a network message initiates its first transmission to the other end, it places a binary 1 in this field. Each subsequent Network transmission sees this value incremented by one. Composite messages have the same sequence number for both parts of the message. Composite messages are assumed to arrive at the receiver in order, with the Odata segment of the Network message first. If the receiving Network service is waiting for a Pdata second segment of a composite message, and another message with a different sequence number arrives, then the Odata message is discarded and it is up to higher level protocols to arrange retransmission of the lost data. Similarly, if the sequence numbers indicate that only the Pdata segment of a Network data unit has arrived, then it will also be discarded.

N-flags: 709 is used to indicate optional N-protocol fields that may be present with the message, and control the segmenting of a Network request into several network messages.

- Segmentation Control bits (reserved)
- Connection data present bit

The high order three bits are used to control segmentation of the message (reserved).

The fourth bit is ON if connect information is included in the message. If such information is present, it will be included in the Odata and removed from the Odata before the Odata is presented to the N-READ or N-OFFER request. Basically, this connect data consists of the PAM that was used to route a message to the receiving application.

When the originating side of an N-connection establishes the connection, it will specify that connect data is present (and include the data) for every message written until a Network level message successfully arrives back to the originator. Thus it is possible that the connect data may be sent several times before an acknowledgement from the other end indicates reception of the connect data and turns the option off. Since

the higher level Data Mover protocols require a protocol "handshake" before the connection is fully open, in practice this means that the N-connect information is only sent until the receiving Data Mover successfully reads a connect message.

The non-originating side of the connection examines this data (if present) and uses it as the route for all information to be returned to the originator. If connect data has already been successfully received when N-connect information arrives, then the receiving network service will examine the new N-connect information and use it as the return route if the value of N-hostcode matches the originally received value (see N-hostcode, below). The purpose of this logic is to permit new new routes to be established if the original one is lost. If a higher level protocol on the originating end detects failure to deliver data, it can issue an N-DISCONNECT followed by an N-CONNECT specifying the same N-ref as the old. PAM information associated with the new N-CONNECT route will be sent to the receiver. Without breaking the connection from the receiver's side, new routing information arrives and is used for all later responses. This permits higher level protocols to gracefully reroute information while maintaining the integrity of the connection.

N-type: 710 contains the type of indication that is to be presented to the receiving Network user. The value of this field is the same low order four bits as is found in the NRBREQ field of the transmitter's request; thus

1 Connect (N-CONNECT) indication
2 Confirm (N-CONFIRM) indication
3 Data (N-WRITE) indication

The other Network requests (N-DISCONNECT and N-STATUS) do not result in message traffic and hence other values will not appear in the N-type field.

N-CONNECT, N-CONFIRM PROTOCOL DATA

FIGURE 6B illustrates the content of the Network Connect subfield of the TPO Network protocol data.

N-OPlength: 711 contains the total number of octets of N-protocol information that is to be found in the Odata. The start of the Odata provided by the user may be found by adding the value of N-OPlength to the starting address of the Odata.

N-OConlen: 712 contains the number of octets to be found in the N-connect field. This separate length field is designed to support the presence of other N-protocol information than N-connect data in the Odata area, If only N-connect data is present, it will generally be one less than N-OPlength .

N-Otype: 713 Indicates that the data following is N-connect information. The following values are valid:

01 - N-connect protocol follows
02 - N-confirm protocol follows; this is identical to the N-connect protocol except there is no PAM.

N-PDsize: 714 In this field, the sender of the Connect data declares the maximum amount of Pdata that it is prepared to receive in a message issued by the sender. Generally this value is a software limitation on the part of the transmitter; it wishes to receive Network messages of a certain maximum size so that it can

anticipate storage requirements before reading data. This value may be either greater than or less than the media message size limitations as specified in the PAM later in the connect data; it is the responsibility of the receiving Network service to send messages no greater in size than the minimum of both fields.

N-ODsize: 715 allows the transmitter to inform the receiver of the maximum amount of Odata that may be sent with any particular network message. The maximum Odata amount is a software limitation that may be independent of the Pdata maximum size or the physical limitations of the connection as specified in the PAM. It is the responsibility of the receiving Network service to send messages no greater in size than the minimum of the received N-ODsize and the local max size.

N-HostDref and N-HostName: 716 and **717** (collectively referred to as N-HostID) accompany the Network Connect field to prohibit Incoming Network Connect Indications from interfering with a new connection. When the offering Network Service receives an' initial Network Connect Indication (the one that completes the initial N-OFFER), it saves the value of N-HostID with the parameters of the Network connection. If a subsequent Connect Indication comes in with the same pair of N-ref's before any N-DISCONNECT has occurred on this side, then it compares the value of N-HostID with the value supplied on the initial connection. If N-HostID does not match, then the message is discarded.

If a Network service that issued. the initial N-connect receives an N-connect message from the network, it is always ignored. The purpose of this field is to prevent connections from being preempted by other hosts that happen to have matching pairs of N-Ref's. Although matching N-ref's will prevent most cases of an inadvertent reconnection at the Network layer, it is not satisfactory in a Data Mover environment for two reasons:

1. OSI standard protocols use the concept of a "reference timer" - an interval in which a given N-ref on a host cannot be reused. This time is specified as being longer than the maximum lifetime of a packet in the network. The high speed networking environment of the Data Mover makes it unacceptable for a service on a known N-ref to be out of action for the maximum lifetime of a message in the network, which could be many seconds.

2. Since a new Network Connect Indication causes all returned data on the N-connection to be rerouted via the route specified in the PAM, an incorrect connect message will not simply be ignored - it can cause loss of the entire Network connection that will require recovery by higher layers of software.

Although the only real requirement for N-hostlD is that there is reasonable assurance that each copy of the Data Mover in the network generates a unique N-hostlD, a specific algorithm is needed. When the Network services establishes contact with the network by establishing its own D-connection, it saves the first D-ref it acquires after initialization.

Physical Address Map: 718 is a data structure that specifies a route to a particular host.

FIGURE 6C illustrates the content of entries in the Physical Address Map list.

Number of octets in PAM list: 720 specifies the entire length of the PAM list.

Number of PAM entries: 721 specifies the number of PAM entries in the list.

First PAM: 722 contains the first PAM entry.

Second PAM: 723 contains the second PAM entry.

Last PAM: 72x contains the last PAM entry

FIGURE 6D illustrates the content of entries in each of the Physical Address Map entries.

Length of PAM: 730 contains the length of this PAM.

Maximum transmission block size: 731 contains the maximum transmission block size for this route.

Maximum transmission rate: 732 contains the maximum transmission rate for this route.

Propagation delay: 733 contains the propagation delay for this route.

TRANSPORT LEVEL PROTOCOL

The present invention contains a method of forming Data Mover messages that include Transport Protocol data for the purpose of communicating with the peer-level Transport layer on the remote side of the Transport Protocol Optimizer network.

The primary functions to be performed by Transport are those of retransmission, flow control, and message segmenting. The Data Mover is designed to work under a wide variety of network configurations, but it is particularly well suited to operate effectively in high bandwidth networks over long distances; and over clean networks as well as over networks experiencing moderate to high rates of packet loss. To accomplish this, the Data Mover is designed to allow dynamic calculations of round-trip times, bandwidth capacity, and rate control and in general, being more adaptive to dynamic network conditions. Assuming there are adequate resources available on both the sending and receiving side, this protocol allows the Data Mover to be extremely efficient in bandwidth utilization. It is designed to operate in either a "closed network" environment, where maximum usage of bandwidth is not an issue (in fact may even be desirable), or in an "open network" environment, where the protocol needs to be aware of co-usage of the network. The dynamic adaptive rate control mechanisms implemented in this protocol allow it to operate in this environment, and co-exist with other applications.

In addition, it must perform a more complex negotiation process than the lower Network level - the OFFER/CONNECT/CONFIRM/READ mechanism. To accomplish this, the transport protocol sends Odata to its partner Transport process, along with the Odata and Pdata provided by the higher level application or Session level software.

The design of the transport protocol is simplified by the fact that the Data Mover Transport does not perform multiplexing. A single Transport process will allocate a single Network connection and enhance the quality of data delivery over that single Network connection. As a result, connection numbers (T-ref's) and other connection identification need not be exchanged as the Network protocol provides the capability to exchange data independently between a large number of tasks.

This chapter is organized into two sections. The first gives a detailed description of the protocol fields present in the Data Mover Transport Protocol. The second section discusses the way that Transport messages are used to provide the services of the Transport Protocol.

TRANSPORT DYNAMIC ADJUSTMENTS

In order to dynamically make adjustments to the Data Mover protocol to meet the conditions of the network, periodic adjustments are made on the following items:

Rate Control
The rate controls are established by the sending side of the Data Mover matching the speed at which the remote Server Application, running on the remote Server, is receiving the data from the remote Transport Protocol Optimizer. The rate at which the remote Server Application is receiving the data is returned in the **TMSGRSPD** value in the Transport Protocol Acknowledgement Subfield (820 in FIGURE 7D).

Latency Time
The latency time represents the additional time incurred for the act of transmitting data over a network. Longer network distances result in higher latency times. The latency times are determined by the sending side of the Data Mover, by calculating the round trip times for the transmission and acknowledgement of messages.

Bandwidth Capacity
The bandwidth capacity represents the total amount of data that can be on the network at any point in time. The Data Mover makes adjustments to its bandwidth capacity value by periodically calculating a new value as the values for "rate" and "latency time" change. The formula used for this calculation is as follows:

Capacity = bandwidth (bits/sec) x round-trip time (sec)

TRANSPORT PROTOCOL FORMATS

Transport protocol consists of two types of fields of Odata. Present in every message sent between Transport services is a Transport Base field, which contains information common to all Transport transactions, and Transport Subfields which contain information specific to a certain Transport activity such as connection, data transfer, acknowledgement, etc. Any number of subfields may be found in a single Transport message.

The types of subfields encountered are:

1. **CONNECT.** This contains information so that the two transport processes can begin a connection and exchange information about the other that will be common throughout the lifetime of the connection. Generally (though not always) this field will be sent with user T-CONNECT and T-CONFIRM requests.

2. **DATA.** This field describes the data that is being delivered to the remote application. It contains such information as the amount of application Odata, block numbering information, segmenting information, and the like.

3. **ACK.** The ACK field is a mechanism for exchanging information about the "state" of the remote transport mechanism. It contains information on blocks received and flow control information. It is present in all messages, and is also sent as an "idle" message if the connection is unused for awhile.

4. **CLOSE** is a special form of the Data subfield that indicates a T-CLOSE has been issued by the application.

5. **DISCONNECT** indicates that a disconnection is to be performed. This may be caused by the application issuing a T-DISCONNECT or a Data Mover detected termination condition (such as user program termination.)

TRANSPORT FIELDS

Transport Base

FIGURE 7A illustrates the content of the Transport Base subfield. It is present in every message sent by Transport service.

TLENGTH: 800 contains the length of all Transport Odata, including the length field itself. Adding TLENGTH to the start of the Transport Odata will give the address of the first octet of Transport application (Session) Odata.

TLEVEL: 801 contains the version of the protocol. This contains either a value of 2 (decimal) for Type2 protocol, or 4 (decimal) for Type4 protocol.

TMSGTYPE: 802 contains the indication type that is to be delivered to the receiving application. Its contents will also dictate the types of subfields that may follow in the Transport Odata. The values that may be placed in this field are:

0 - **IDLE**. This transport message is only used to verify and report the continued existence of both Transport processes. No indication is to be delivered to the Transport application as a result of this message.

1 - **CONNECT**. The Transport application has issued a T-CONNECT request. A data field associated with this message will contain the data supplied with the T-CONNECT. A Connect transport subfield will be present. A Connect Indication will be placed in NRBIND of the receiving application's T-OFFER request.

2 - **CONFIRM**. The transmitting Transport application has issued a T-CONFIRM request. A data field associated with this message will contain the data supplied with the T-CONFIRM. A Connect transport subfield will be present. A Confirm Indication will be placed in NRBIND of the receiving application's T-READ request.

3 - **DATA**. The transmitting Transport application has issued a T-WRITE request. A data subfield will be present. A Data Indication will be placed in NRBIND of the receiving application's T-READ request.

4 - **EXPEDITED DATA**. (Reserved).

5 - **CLOSE.** The transmitting Transport application has issued a T-CLOSE request. A data subfield will be present. A Close Indication will be placed in NRBIND of the receiving application's T-READ requests. Only DISCONNECT and IDLE message types may follow this Transport message on this connection.

6 - **DISCONNECT**. Either the application or Transport service has initiated a T-Disconnect. A Disconnect subfield will be present and a Data subfield may be present. A Disconnect Indication will be placed in NRBIND of the receiving application's T-READ request. This is the last Transport message that will be sent in this direction by Transport service.

C nnect Subfield

FIGURE 7B illustrates the content of the Transport Connect subfield that is located in a TPO message. This subfield contains the information needed to correctly exchange information between the two Transport processes for the lifetime of the connection. It covers such things as timeout thresholds, segmenting information, and maximum user block sizes.

TSLEN: 803 gives the length of the connect subfield, including itself. Adding the value in TSLEN to the address of the Connect subfield will give the start of the next Transport subfield.

TSTYPE: 804 identifies the type of subfield and hence the nature of the information to follow. A Connect subfield has a 3 (binary) in this octet.

TMSGITO: 805 is used to inform the receiver how often it should send "idle" messages back to the transmitter to keep the connection open. An idle message is then sent whenever TMSGTO seconds have elapsed without the Transport process having anything to send. The reception of an idle message alerts the remote side that blocks may have been lost in transmission. Such an idle message will contain an ACK subfield which informs the other side what information has or has not been received.

Each side can have a different threshold for its idle timeout. Note that each side tells the other how often the messages should be sent; a side has no control over the interval in which it should transmit idle messages.

In order to process such idle timeouts, the Transport process will have to continue interval timing even though no work is in progress. If a Transport process does not receive an idle message or other response within a larger interval (referred to as DEADTO), then it will consider that communications have been lost and abandon the Network connection in use.

TMSGTTO: 806 In this field, the two processes exchange the number of seconds the other should wait for an attempt to re-establish a Transport connection over a different path. The connecting side sends the calculated value:

(# PAM entries * CONTO) + DEADTO

The offering side sends: DEADTO.

CONTO and DEADTO are connection timeout values assigned by Transport service; their meaning is explained in detail later in this chapter. "# PAM entries" is simply the number of usable paths that have been supplied to reach the destination by the user of Transport service.

This field is used when it is associated with an initial T-CONNECT to a Transport user who has issued a T-OFFER. If the offering side must wait for Alternate Path Retry to take place by the T-CONNECTING side, it gives the maximum amount of time that the offering side should wait for a connect attempt to succeed.

TMSGBLKO: 807 is used to pass the NRBBLKO supplied by the user with a T-CONNECT or T-OFFER request. This field is the largest size block that the Transport application will wish to send for the lifetime of the connection. This field will be forwarded to the receiving user along with the indication type of the message.

Transport does not place an inherent limit on the size of data blocks moved; however, transport applications may wish to do so. Transport software can use the values in BLKO and BLKI to better manage buffer and network resources.

TMSGBLKI: 808 contains the maximum input Pdata size acceptable to the Transport application as specified in NRBBLKI of the T-CONNECT or T-OFFER request.

TLCM: 809 contains the Least Common Multiple of the number of bits required to fill a machine addressable word on the transmitting host. It can be used to ensure that when user data is segmented that the segments always end on a word boundary when received. See the section on "Segmentation" for more information on the use of this field.

TMSGACKC: 810 contains the "Ack Credit" of the Transport process that is sending the Ack subfield. It specifies how many blocks the transmitting process is prepared to have outstanding before the receiving process must return a message containing an Ack subfield.

If a 0 or 1 is in this field, then the transmitter of this Ack subfield desires that an Ack subfield be returned every time a message is received by the remote transport process. If the remote process has no data ready to be sent on which an Ack subfield may be "piggybacked," then the remote process will have to generate an "idle" message to immediately respond to the incoming message. This low value should only be used by highly constrained Data Mover implementations that have very limited buffer space or only function in a simple, one-thing-at-a-time communications environment.

If a value greater than 1 is in this field, then the Transport process receiving this subfield may allow that many incoming data messages to arrive without generating a responding message with an Ack subfield. A value of 2 or greater is particularly valuable in inquiry/response applications because the block containing the inquiry can be acknowledged by the transmission containing the response; the response can be acknowledged in the block containing the next inquiry, etc.

Transport Protocol Data Subfield

FIGURE 7C illustrates the content of the Transport Protocol Data subfield. This subfield describes the Odata and Pdata supplied by the transmitting Transport application.

TSLEN: 811 gives the length of the data subfield, including itself. Adding the value in TSLEN to the address of the Data subfield will give the start of the next Transport subfield.

TSTYPE: 812 identifies the type of subfield and hence the nature of the information to follow. The Data subfield may have one of two values:

- 4 - indicates that the data herein is the last segment (or only segment) of a block of data provided by a Transport user.
- 7 - indicates that the data is a segment other than the last of a user provided block.

Therefore a large block provided by a transport user would consist of several blocks with TSTYPE 7 followed by a last block with TSTYPE 4. Since each segment of the user's block has its own physical and logical sequence number, reassembly of the user data on the receiving side is quite straightforward.

TMSGLNO: 813 represents a thirty-two bit unsigned binary quantity that increases by one each time a

new segment of transmitted data (a Transport Protocol Data Unit) is first transmitted on the network. This number will be the order in which transport messages and indications will be delivered and data reassembled for the receiving application. Therefore, data is ordered by Logical number in the order it was provided in write-type transport requests.

The purpose of this field is to permit flow control from transmitter to receiver and reassembly of incoming data by the receiver.

The first block of transport application data (which will be T-CONNECT data) will have a TMSGLNO of 1. This number will increase by 1 until 2G-1 blocks have been sent; then it will wrap around to 0 and continue incrementing.

TMSGPNO: 814 is a thirty-two bit unsigned binary quantity that is incremented by one each time a Transport message is given to Network service for transmission over the network. This field is used to coordinate the acknowledgements of data received.

The first transmission by Transport service (which will contain the T-CONNECT data) will have a TMSGPNO of 1. This number will increase by 1 until 2G-1 N-CONNECT or N-WRITE requests have been made; then it will wrap around to 0 and continue incrementing.

Transport Protocol Acknowledgement Subfield

This field is used to report to the opposite party the "state" of the transport process - what has been sent before and what has been received. By examining this field, the other transport party can determine what has been successfully sent, whether the other party can accept more data (flow control), and if blocks were recently sent which did not arrive at the receiver (lost block detection).

This field is present in every Transport message, whether or not the Ack credit is exhausted. This allows the transmitter of information to free storage for successfully transmitted blocks on a more timely basis, and improves network performance in environments where acknowledgements can be lost.

FIGURE 7D illustrates the content of the Transport Acknowledgement subfield located in the Transport portion of a TPO message.

TSLEN: 815 gives the length of the acknowledgement subfield, including itself. Adding the value in TSLEN to the address of the Ack subfield will give the start of the next Transport subfield.

TSTYPE: 816 identifies the type of subfield and hence the nature of the information to follow. An Ack subfield has a 2 (binary) in this octet.

TMSGMAXO: 817 contains the highest Physical Block number (thirty-two bit number) that has been sent by this side of the Transport connection. In most cases, where a Data subfield is also included in the message, this field will be equal to TMSGPBN in the data subfield. In messages without a Data subfield (typically Idle messages), it will contain the highest value of TMSGPBN introduced on the network.

When received in an "idle" message, this allows the opposite Transport process to detect the fact that an extra block or blocks has been sent by the remote party but not received locally. The local transport process can then generate a Transport message of its own with an Ack subfield that negatively acknowledges TMSGMAXO back to the last block successfully received.

TMSGMAXI: 818 contains the highest value Physical Block Number (thirty-two bits) that is known to have been transmitted by the remote Transport process. It will contain the highest value of TMSGMAXO known to have been correct on an incoming message, whether or not the data was successfully received. This highest known value is returned back to the remote transport process so it can use the Ack/Nak information in TMSGACKI to retransmit data.

TMSGACKI: 819 consists of sixteen bits of Ack/Nak information. It informs the receiving Transport process A of the status of the last sixteen blocks of information that Transport process B knows A has tried to send.

The high order bit of this two octet field contains the Ack/Nak status of the Physical Block Number in TMSGMAXI. If this bit is off, then process B has successfully received Physical block number TMSGMAXI; if on, then it is not been successfully received and that particular block should be immediatedly scheduled for retransmission (with a new, higher Physical Block Number). The next most significant bit contains the Ack/Nak status for block TMSGMAXI-1; the next for block number TMSGMAXI-2 and so forth until the oldest block covered in this field, TMSGMAXI-15, is reached.

This bit significant information is designed to be redundant; in this manner, the loss of one or more messages containing acknowledgment subfields is not sufficiently serious that the processes have to "time out" and exchange idle messages to resynchronize themselves.

Therefore, a transport process receiving an Ack subfield need only process the Ack/Nak information for those blocks not reported upon in the previous Ack subfield received. Since retransmitted blocks acquire a new Physical Block Number (while retaining their original Logical Block Number), multiple retransmissions of the "same" block do not present any problems.

If a sufficient number of Ack subfields were lost that TMSGMAXI-15 is greater than TMSGMAXI of the previous Ack subfield received, then all blocks without definite positive acknowlegement below the newer TMSGMAXI should be retransmitted - the Nak'ed blocks between new TMSGMAXI and new TMSGMAXI-15 and the unreported blocks between new TMSGMAXI-15 and old TMSGMAXI, as well as continuing retransmission of all Nak'ed blocks at or below old TMSGMAXI. A receiving Transport process must be prepared to discard any incoming blocks with a Logical Block Number of a block already received successfully.

When the Transport process is just beginning, not all sixteen bits will contain the status of an actual block. Bits showing the "status" of Physical Block Numbers below 1 (the first message) should be set to zero.

TMSGRSPD: 820 contains the rate of delivery of data to the remote application.

Idle Messages

An "idle message" is defined as a message containing no application data which is exchanged between the two transport processes. It contains only a single Transport protocol subfield, the Ack subfield. Its purpose is to inform the remote transport process of the continued presence of the connection, and to permit the retransmission of any data that may have been lost.

There are three main circumstances where a Transport process should send an idle message:

1. When data is arriving from the remote Transport process and the Ack Credit supplied with the last Ack

subfield provided by the remote party is exhausted.

Furthermore, the local Transport process has no data to send. In "one-way" applications such as file transfer, the receiving Transport process will generate an idle message every "ack credit" times an incoming data message is received.

2. When the local Transport process has not had occasion to transmit anything (including another idle message) for the time interval that was specified by TMSGTO in the last Connect subfield received from the remote transport.

   Since both sides of the connection are performing this transmission, they can detect loss of communications with the other side of the connection by the failure to receive idle messages for an extended time. In addition, a burst of errors or a small Ack credit may cause both sides to think all is well when in fact one or more recent transmissions may have been lost. The exchange of timed idle messages will inform the transmitter that data was lost so that retransmission may take place.

3. When the flow control mechanism implemented by TMSGPROC must be used to tell a transmitting process that a "buffer full" condition in the receiver has been relieved.

   If a pair of applications work so that the transmitter is faster than the receiver, then the receiving Transport process' buffers will fill and a value in TMSGPROC will be returned so that the transmitting Transport process will introduce no new data on the network. At a later time, the receiving application will free buffers on the receiving side by accepting data. When the number of buffers holding data drop below a threshold determined by the receiving Transport process, then it will send an idle message back to the transmitter with a higher TMSGPROC value. The transmitting Transport process can then send several more blocks and the process will continue.

Transport Protocol Disconnect Subfield

FIGURE 7E illustrates the content of the Disconnect subfield located in the Transport portion of a TPO message. The Disconnect Subfield is sent whenever a Transport process is unilaterally shutting down and wishes to inform the remote Transport process of the fact. It will be the last message transmitted by the side initiating the Disconnect. Any subsequent messages arriving at the Disconnecting side will be discarded. The purpose of this subfield is to inform the Disconnected (remote) Transport process of the reason for the disconnect.

TSLEN: 821 gives the length of the disconnect subfield, including itself. Adding the value in TSLEN to the address of the Disconnect subfield will give the start of the next Transport subfield.

TSTYPE: 822 identifies the type of subfield and hence the nature of the information to follow. An Ack subfield has a 1 (binary) in this octet.

TMSGWHY: 823 gives the sixteen bit error code that should be placed in NRBSTAT of the next T-READ request to be provided by the Transport application. Consequently, it should be an error code that indicates to the application error conditions detected in the remote host (such as remote Transport application abnormal termination).

If the message containing the Disconnection subfield is the result of a T-DISCONNECT request issued by the remote Transport application, then TMSGWHY will contain a binary zero. The next local T-READ

request will complete with a Disconnect Indication and an NRBSTAT of zero, indicating a normal, successfully executed Disconnect.

Note that since the Disconnect message is not acknowledged, no further communications will take place if the message is lost. As a result, some sessions that would end with a normal message may end with NRBSTAT 2400 (communications lost) instead of the intended error code.

TRANSPORT MESSAGE CONSTRUCTION

All communications between Transport processes will contain one or more of these Transport subfields. The following table shows what sort of subfields can be expected with the various types of Transport messages and indications.

Message Type	Connect	Data	Ack	Disc	Segment
CONNECT	Req'd	Req'd	Req'd	Never	Never
CONFIRM	Req'd	Req'd	Req'd	Never	Never
DATA	Option	Req'd	Option	Never	Option
CLOSE	Option	Req'd	Option	Never	Option
DISCONNECT	Never	Option	Option	Req'd	Never
IDLE	Option	Never	Option	Never	Never

This chart contains several elements that should be noted:

1. The Connect subfield may optionally be included with most normal Transport messages. It may be sent at any time by a Transport process that wishes to work with new block size, timeout, or segment size values. Most typically it would be used by the connecting Transport process when a Network connection was lost and a new one was established. The offering Transport process would then get a Connect Indication on its N-Read, along with a Transport message containing normal data and possibly a new Connect subfield to reflect changing parameters in the new Network connection.

2. Segmentation of data (Data messages without the End flag indicating that more data is needed before a T-READ is to be marked complete) is only permitted with Data and Close indications. It is not supported while the connection is being opened, as maximum Transport block sizes have not yet been negotiated.

3. Although it is a very good idea to include an Ack subfield along with a Transport message containing data, its presence is not required.

4. An idle message may contain a Connect subfield, and it may also not contain an Ack subfield. (However, there seems to be no benefit and several problems possible if it is omitted). If neither subfield is supplied, then the only effect of the Transport message is to reset the Connection lost timeout counter (see "Transport Timeouts" below).

5. Disconnect messages may contain Data subfields to describe data associated with a normal T-DISCONNECT service request. They may also contain Ack subfields, although these can be ignored by the receiving Transport process. A Disconnect subfield will only be seen with a Disconnect indication.

6. Incoming messages that violate these rules may be treated as "protocol errors" and their presence ignored.

TRANSPORT TIMERS AND TIMEOUTS

All versions of transport protocols need to continuously monitor three time intervals that are used to monitor the state of the connection.

IDLETIME - The time interval since the last message was sent to the remote application. If IDLETIME increases to the point where it equals TMSGITO as last received from the remote application, then an idle message must be sent to the remote Transport process.

DEADTIME - The time interval since the last message of any description was received from the remote transport process. If this interval reaches a maximum time interval, TMSGTTO received from the remote side, then communications on the connection is assumed to be lost and another path will have to be retried.

CONTIME - An abbreviated version of DEADTIME that is used immediately after a new N-Connection has been established. The new N-connection could take place either during Transport connection establishment or just after a new path has been selected due to loss of communications. It is the time interval since the N-CONNECT was issued. If a maximum time interval, CONTO, elapses with no response of any description returned in response to the Transport message sent with the N-CONNECT, then this path is assumed to be not functional and another path will be tried. The value of CONTO should be large enough to allow several attempts to send the connect message and receive a confirmation. The connect is attempted every IDLETO until a response is received or CONTO is exceeded.

If either end of a Transport connection has its DEADTIME reach DEADTO, then it will conclude that communications are lost. Alternate Path Retry will then take place or the connection will be terminated.

TRANSPORT CONNECTION SEQUENCES

Transport Connection Messages

FIGURE 7F illustrates the Data Mover transport connection sequence. The Transport Connection process basically takes place in the T-OFFER - T-CONNECT - T-CONFIRM sequence discussed in the user

documentation. The only thing that is worthy to note at this point is that the "ack credit" field sent with the Connect protocol data will dictate if an Idle message will be sent by the offering side in response to receipt of a Connect Indication. If the Ack Credit is 0 or 1, an Idle message will be immediately returned. If it is 2 or higher, then it can be returned with the Confirm Information unless the offering application takes more than IDLETO seconds to generate a T-CONFIRM response.

It is also worthwhile to note that the connection is "half duplex" until the T-CONFIRM is sent or the Confirm Indication is read. After that time, T-WRITEs may overlap in any way on a full duplex connection.

The other considerations are what happens when a process issues a T-CONNECT and no response is returned from the other side. The full responsibility is with the connecting party, as it proceeds as follows:

1. When the T-CONNECT is issued, an Idle message is sent after the connect every IDLETO seconds. The connect is also resent every IDLETO, since it may be lost. If any message containing a Confirm or Idle with the proper N-refs arrives, then the path is considered a good one. Alternate path retry will subsequently only take place after more than DEADTO seconds have elapsed (see later in this chapter).

2. If CONTO seconds have passed since the T-CONNECT was issued with no response of any kind, then it is assumed that the path selected is inoperable. Alternate Path Retry begins. An N-DISCONNECT is issued to terminate and clean up the existing Network connection. The next path in the list of PAM's provided to transport is selected. A new N-CONNECT with the new PAM is issued. The clock on this new connection will run as described in steps 1 and 2. APR may be invoked immediately if network returns a status indicating that the selected path is unavailable (adapter inoperable). If all paths receive this status, then the T-CONNECT attempt can be aborted early.

3. If there are no more PAM's to be examined, then the total time since the first T-CONNECT was issued is examined. If the elapsed time is less than DEADTO, then we go back to step (1) with the first PAM in the list all over again. Thus on any connection, continual attempts to re-Network Connect and send a T-Connect will take place for at least DEADTO seconds, distributing the connections over all possible paths as equitably as possible.

It is possible that temporary connection failure may have taken place because the offering Data Mover was not able to return an Idle or Confirm message within CONTO seconds. If that was the case, then the offering side will ignore subsequent Network Connect indications and issue an N-CONFIRM using the route it first received the Network Connect Indication. This may result in each side of the connection using a different Network path, but this is an acceptable situation.

CONNECTION TIMEOUT AND ALTERNATE PATH RETRY

One of the capabilities provided by the Transport Service is the ability to divert traffic over a totally independent Network connection without causing a loss of service to the Transport user. If Transport detects that all communications have ceased on the existing Network connection, it will N-Disconnect the failing connection and attempt to N-Connect using the next PAM provided in that list of PAM's passed to Transport at T-Connect time. If the N-Connection succeeds, then the transport connection will continue as normal. If it does not, then all PAM's in the list will be tried until the end is reached. If the last entry fails, then the Transport user will receive a Disconnect Indication indicating that the connection has been lost.

Alternate path retry is basically the responsibility of the Transport entity that first initiated the connection by accepting a T-CONNECT request from the user. It is the responsibility of this party to try all possible Network connections to the remote Transport process. It is the responsibility of the other (offering) side to simply wait until sufficient time has elapsed for all paths to be tried, or until input arrives from the connecting side. The explicit actions taken are discussed below:

- **OFFERING SIDE**. If a transport process originally began with a T-OFFER, it will continue to transmit Idle messages every IDLETO seconds (if it is not sending data messages), and will not take other action until TMSGTTO seconds expire with no response of any type from the remote side. At that point, the connection has been lost; the Transport Service informs its local user of the loss with a 2400 NRBSTAT error code.

  If a Network message containing a Connect Indication does arrive during that time, it considers retry to have succeeded. It then sends a Transport message (either an Idle or some data) using the N-CONFIRM service. Transport will continue to send N-CONFIRM responses until a message with a normal Data Indication arrives. At that point, it will N-WRITE all subsequent Transport messages.

- **CONNECTING SIDE**. The initiating Transport service will continue to send Idle messages every received TMSGITO seconds and will not take action until DEADTO seconds have elapsed with no response whatsoever from the remote Transport Service. Then Transport will N-DISCONNECT the original Network path and N-CONNECT on the next PAM entry. The N-CONNECT data will contain a Transport Idle message or possibly Transport Data.

  Once the new path is selected, the Transport process will continue to try this new path for CONTO seconds, sending Idle messages with an N-CONNECT request every IDLETO seconds. During this whole time, it leaves an N-READ outstanding; if data arrives with a Network Confirm Indication within CONTO seconds, it considers the new path to be good and will not forsake it until a subsequent DEADTO seconds have elapsed without a Network message arriving from the remote end.

  If no data is received from this new path within CONTO seconds, then the next path in the list of PAM's is selected. Once again an N-DISCONNECT and a new N-CONNECT is issued and the Transport service waits CONTO seconds for the connection to complete. Transport will continue to try new paths until all paths in the PAM list originally provided with the T-CONNECT have been tried with no response whatsoever from the remote side. Transport must have the logic to "wrap around" a PAM list and retry earlier paths in the list if successful alternate path retry took place earlier in the history of the connection.

FIGURE 7G illustrates the connection re-establishment sequence that occurs after a network outage. Both sides lose proper connection at about the same time. The offering side is prepared to receive a new connection at any time; The connecting side waits DEADTO seconds for a response, then tries a connection on a new Network path.

Message Size Negotiation

One of the things that distinguishes the Data Mover from other ISO based protocol is the fact that Protocol Data Unit sizes are not fixed in the network but are separately negotiable between different machines. This is considered quite valuable in networks where personal computers and supercomputers may coexist, and also because of the fact that block size is the biggest performance factor at megabit transmission speeds due to the overhead introduced by all operating systems whenever I/O to a network is performed.

It is the responsibility of the Transport service to determine the maximum size of data that should be sent over the network at any one time. Most of these functions are performed by the Network service. Four factors determine the maximum amount of data that may be sent with any single transmission initiated by the Transport service:

1. Any physical limitations on size posed by the network media.

2. The maximum amount of data acceptable by the remote Network service.

3. The maximum amount of data acceptable by the local Network service.

4. The "voluntary" declaration of transmission maxima for input and output by the Transport users.

Although the Network service will determine the actual maximum sizes that may be transmitted based on these four factors, it is the responsiblity of the Transport service to observe these size limitations and segment user's data accordingly. When the connection is first being established (T-CONNECT and T-CONFIRM requests are being serviced), the user may not provide more data than can fit into a single Network Write request; after that point, data sent with T-WRITE and T-CLOSE requests must be segmented by the Transport Service and reconstructed on the receiving side.

Segmentation of User Data

The Data Mover resembles "standard" protocols in that there is no inherent limit on the amount of data that a Transport user may send and receive with a single request. It is the responsibility of the Transport service to break a large TPDU into several "segments" that are suitable for transmission over the media controlled by the Network Service.

The Data Mover accomplishes this in a conceptually simple manner: a caller's data is broken into "segments", each of which is small enough to be delivered by the Network service. Each segment contains a new, incrementing Logical Block Number as discussed in the previous section. The last segment of a TPDU contains an "end" flag which indicates that the TPDU is to be delivered to the user. When all segments for a TPDU have arrived, they may be presented to the receiving Transport user.

Segmentation of data is only supported with the WRITE and CLOSE services. Data sent with CONNECT, CONFIRM, and DISCONNECT requests must be smaller than or equal to the maximum block size determined by the Network service. Also it should be noted that segmentation applies only to the Pdata provided by the Transport user; Odata is not segmented.

TRANSPORT CLOSE FACILITY

The CLOSE facility is designed to be functionally compatible with the Graceful Termination facility defined by the NBS protocols. It was not present in the original Transport Protocol because of the continued resistance to graceful termination at the Transport level by the ECMA and ISO standards groups. NSC saw fit to put it in because of NBS and the fact that existing user applications can be much more easily written without having to be careful about their own termination. As will be seen, some of the end cases with CLOSE can be unpleasant, and it is an imposition to require this logic of the Presentation service.

Either party of a connection may issue a T-CLOSE request at any time that a T-WRITE is valid. Once a T-CLOSE is issued to the local Data Mover, only a T-READ or T-DISCONNECT may be issued against that application by the user. The data provided with a T-CLOSE may be of indefinite length, just like the T-WRITE; thus the Transport Service must perform block segmenting on a T-CLOSE request.

When it is the appropriate time to present the block of data written with the T-CLOSE to the receiving appliction, it is presented with a Close Indication in the NRB. At that time, the receiving application knows that it will receive no further data (with the possible exception of a Disconnect Indication) from the application that originally issued the T-CLOSE. The receiving application should continue to leave a T-READ outstanding in the event that a Disconnect Indication may need to be presented.

After the Close Indication is received, then the receiving application may continue to T-WRITE blocks of data to the application that sent the T-CLOSE indefinitely. The T-CLOSE transmitter will continue to issue T-READ requests to accept the data. When the T-CLOSE receiver has reached the point of sending all relevant data to the transmitter, it will issue a T-CLOSE of its own, which may contain data in the same

manner as the previous T-WRITEs. When the T-CLOSE is marked complete, then the connection is terminated from the receiver's point of view. If a T-READ is outstanding at the time that the second T-CLOSE is issued by the receiver, then it will terminate with no indication in NRBIND, 0 status in NRBSTAT, and no data received.

When the side that transmitted the original T-CLOSE receives a Close Indication, then the connection is fully closed from its vantage. No other requests against that connection, including a T-DISCONNECT, will be accepted.

CLOSE message sequences

CLOSE is a pretty straightforward piece of code were it not for the concern of being really sure that the last message or acknowledgement has been properly sent and received. The normal case is shown as follows:

Suppose that the acknowledgement of the second Close Indication were lost. At that point, the party issuing the second T-CLOSE could not be sure that information had properly arrived on the other side. Normally this is accomplished in the following manner:

1. The side that initiated the close successfully receives the Close Indication and its associated data. It presents the data to the user and issues an N-WRITE containing an ACK field to acknowledge the data. Note that it may have to force an acknowledgement if the Ack Credit is sufficiently high.

2. After the ACK is transmitted, this side sets IDLETO to DEADTO and remains present for DEADTO seconds before the connection is forgotten. If any message with the appropriate N-refs arrives during this interval, this side automatically generates a message with proper ACK information in response. If nothing arrives, then it does not transmit on the network.

3. After DEADTO seconds have elapsed, the Transport service on the side that issued the original T-CLOSE frees the resources associated with the connection and allows the N-ref's to be reused.

4. The transport service on the side that issued the second T-CLOSE may exit as soon as the acknowledgement to the Close message is received.

SESSION PROTOCOLS

The present invention contains a method of forming Transport Protocol Optimizer messages that include Session Protocol data for the purpose of communicating with the peer-level Session layer on the remote side of the Transport Protocol Optimizer network.

Session service, as implemented in the Data Mover, provides essentially the same capabilities as Transport service. The major difference between the two is that Session provides a considerably simpler means of identifying and connecting to applications that wish to perform network activities. To accomplish this, Session service actually performs two Transport connections during the lifetime of a single Session connection; one to determine the location of the remote application and a second to perform data transfer between the two applications.

THE SESSION MANAGER

FIGURE 8F illustrates the establishment of a communication link between two Transport Protocol Optimizers. The Transport Protocol Optimizers establish a connection during the TPO initialization process that remains as a persistent connection.

The purpose of the Session Manager element is to provide a unique reference number (N-ref) for each offering TPO connection, and returning it to the TPO from which a connection request is received. This reference number is used on all subsequent messages destined for this connection.

During TPO initialization, the Session Manager issues a T-OFFER, using a well-known network reference number. When the TPO issues the S-OFFER request, the Session Manager receives control. It places the eight character application name (SESSION1) in an internally maintained list of applications that have "advertised" for service.

Connection establishment begins when the Session Manager's T-OFFER completes with a Connect Indication. The Odata associated with the Connect Indication contains information supplied by the initiation of an S-CONNECT request from the peer Transport Protocol Optimizer. This Odata contains the application name of the process to be connected to (SESSION1), along with the application and host name of the connecting Transport Protocol Optimizer.

The session manager responds to the connecting party by issuing a T-CONFIRM with the results of the Session "match." If the application named by the connector is unknown to the Session Manager, then failure will be reported in the T-CONFIRM Odata. If the application is known, then the entry is removed from the session manager's list of offered application names. N-ref's are assigned for the local offering Transport Protocol Optimizer. A T-OFFER is issued on the behalf of the local offering Transport Protocol Optimizer by the Session manager. The N-ref's that should be used for the connection are supplied to the connecting Transport Protocol Optimizer by the T-CONFIRM request issued by the session manager.

When the remote Transport Protocol Optimizer that is processing the S-CONNECT request has the Confirm Indication returned from the Session Manager, it issues a T-DISCONNECT to free the Session Manager. The connector then issues a T-CONNECT to the N-ref specified in the Confirm data. Finally, the "direct" connection proceeds between the two Transport Protocol Optimizers.

SESSION SERVICE

Once the connection is established to the remote party, then Session service 'drops out" in the sense that a Session service call simply invokes the equivalent Transport service call. A rudimentary Session protocol is included at this point to identify protocol levels, etc.

SESSION MANAGER PROTOCOL

Session Protocol is the Odata that is included by Session service for communications purposes. It consists of a base field that is present in all Session messages, as well as subfields that are used to communicate with the Session Manager.

Session Manager Protocol Base Field

FIGURE 8A illustrates the content of the Session Manager Base subfield. As with the Transport protocol it contains a header with total length and version number followed by a protocol type indicator.

SLENGTH: 1000 gives the length of all Session protocol data that is in the message, including the SLENGTH field itself. Adding SLENGTH to the start of the Session Odata will give the first octet of Presentation Odata to be delivered to the Session application.

SLEVEL: 1001 gives the version number of the protocol in use. Currently, this will have the value of 2 (binary).

STYPE: 1002 Strictly speaking, the Data Mover does not use one session protocol - it uses one of several, depending on the events at hand. This field identifies the "subtype" of session protocol that is contained in this message.

STYPE=0 indicates that this message is part of a direct communication between Session applications after the connection has been established. In that case, only the base field is present; no subfields are defined.

STYPE=1 indicates that a dialogue between Session Service on one end and a Session Manager at the other is taking place. Information on application names and addresses is being exchanged. STYPE=2 and STYPE>2 are reserved.

Session Manager Connect Protocol

FIGURE 8B illustrates the content of the Session Manager Connect subfield, including the Session Manager Base subfield. This information is sent on the behalf of an application that has issued an S-CONNECT. It is a request for a Session Manager to complete a connection to an S-OFFERing application.

Four alphanumeric eight character strings are supplied to match sessions and supply information about the connecting party. All transmissions of this character data over the network must use the ASCII code. Any ASCII character values are acceptable for application and host names.

SPNAMED: 1009 contains the name of the application that is to b connected to. It is the character information supplied in NRBCONN2 on the S-CONNECT request, and must match a string supplied in NRBCONN2 of some outstanding S-OFFER request in order for the connection to complete

SPHOSTD: 1010 The name of the host that the connector wishes to address. This field should match the name of the host assigned during Data Mover generation or initialization. If this field is being generated to perform an S-CONNECT, it should contain the host field in NRBCONN1, translated into ASCII code.

SPHOSTS: 1011 contains the generated name of the host from which the connect request is originating. It is not needed to complete the connection; its intended use is to permit simpler security and accounting checks of potential connections and to make operator displays of outstanding sessions more informative.

SPNAMES: 1012 contains a Data Mover generated name for the application program. The manufacture of this name is system dependent and should be some value which is "unique" at any one time to that computer system, such as jobname, Logon ID, account number, etc. This field is not needed to complete the connection; its intended use is to permit simpler security and accounting checks of potential connections and to make operator displays of outstanding sessions more informative.

Session Manager Confirm Protocol

FIGURE 8C illustrates the content of the Confirm subfield, including the Session Manager Base subfield. This information is sent on the behalf of an application that has issued an S-CONNECT. It The confirm protocol is used when the Session Manager wishes to return information back to the connector. It will inform the connector whether the connection had a matching offer or not, as well as giving information on the route to be followed to reach the offering application.

The response to a connect request comes in one of three sizes depending on the nature of the response.

1. If the connection did not succeed, a reason code is returned that details the reason for the failure. In this case SPLEN=5.

2. If the connection did succeed and the PAM used to connect to the Session Manager is also suitable for connection to the user application, then the N-ref of the application is returned and the connection application can use the Session Manager PAM and the application N-ref to establish the application to application connection

3. If the connection succeeded, but the PAM used to connect to the session manager is not appropriate to connect to the application, then the Session Manager can return a complete list of PAM's, suitable for passing directly to Transport Service with the application to application T-CONNECT request.

The meanings of the various fields are detailed below.

SPFFLAGS: 1018 details the following information.

Su — If on, this indicates that the connection succeeded in matching an OFFER maintained by the Session Manager. If this bit is on, then the SPFNREF, SPNAMED, and SPHOSTD fields will be supplied and the protocol length fields will reflect this fact. The SPFWHY field will contain zero, indicating a successful connection.

SPFWHY: 1019 contains a binary zero if the connection succeeded, and the reason for the failure to connect if the connection failed. In the event of failure, this sixteen bit reason code will be placed in NRBSTAT of the application's S-CONNECT request to indicate the reason for failure. Most often this error will simply indicate that the offered party is not present; however, exceptional conditions or user exits may return other codes or installation dependent codes back to the connecting user through this mechanism.

SPFNREF: 1020 contains the N-ref of the application that is to be directly connected to. It should be supplied in NRBCONN2 of the T-CONNECT for the connection to succeed. This field must be present if the Su bit is on.

SPFNAMED: 1021contains the name of the user program or task that accepted the connection. The generation of this field is system dependent and should contain a "unique" identifier such as jobname, Logon ID, etc. The value of this field is not needed for establishment of the connection. It could be used as a check to ensure that the correct party is connected to and to make operator information displays more informative .

The field consists of 8 octets of character ASCII data.

SPHOSTD: 1022 contains the name of the host computer that actually accepted the connection. Generally, this will be identical to the SPHOSTD value provided with the Session Manager Connect information. However, if the connection was made using host groups or NTXADM, it will be the ultimate destination host.

The value in the SPHOSTD field is not needed to complete the connection. It can be used to make operator information displays more informative and to check for the possiblity of exchanging information with undesirable hosts.

The field consists of 8 octets of character ASCII data.

PAM list:

Following the Session information in the response is an optional PAM list, in the same form as is provided to a T-CONNECT request. This PAM is returned in a form that can be directly used to establish communications with the application that issued the S-OFFER.

This field can be produced by modifying the incoming PAM that T-connected to the Session Manager, or it can be produced by generating a list of routes between the two applications based on the Session Manager's knowledge of the network.

Session Manager Disconnect Protocol

No Odata is associated with a Disconnect request to the Session Manager. The Session Manager will re-offer itself as soon as the first connection is terminated for whatever reason.

SESSION SERVICE PROTOCOL

Session Service Protocol is the Odata that is included by Session service for communications purposes after the two Session applications have established a direct T-connection between one another. It consists of only a base field that is present in all Session messages. No subfields are present, as Transport provides all the services that are needed in this version of Session protocol.

Session Service Protocol Base Field

FIGURE 8D illustrates the content of the Session Service Base subfield. As with the Transport protocol it contains a header with total length and version number followed by a protocol type indicator.

SLENGTH: 1003 gives the length of all Session protocol data that is in the message, including the SLENGTH field itself. Adding SLENGTH to the start of the Session Odata will give the first octet of Presentation Odata to be delivered to the Session application.

SLEVEL: 1004 gives the version number of the protocol in use. This will have the value of 2 (binary).

STYPE: 1005 contains a binary zero, indicating that this session protocol is intended for direct application to application connections. Only the base field is present when STYPE=O.

Abbrieviated Session Protocol field

Normally, there is little or no session protocol to be sent in a TPO non-connect type message. Optionally, an abbreviated form of session protocol can be used, as illustrated in FIGURE 8E.

Application Messages

The present invention contains a method of forming Transport Protocol Optimizer messages that include Application Protocol (i.e. Packet Driver) data for the purpose of communicating with the peer-level Application layer (i.e. Packet Driver) on the remote side of the Transport Protocol Optimizer connection.

FIGURE 9 illustrates the format and contents of the TPO Packet Driver application data. It consists of the following fields:

Routing Header: 1200 contains the Routing Header (see FIGURE 9A). There is one Routing Header in each Packet Driver buffer, and is located at the start of the buffer.

Message Header 1: 1201 contains the Message Header (see FIGURE 9B). There is one Message Header for each intercepted IP packet data in the Packet Driver buffer. The Message Header immediately precedes the associated IP packet data.

Data contents 1: 1202 contains the data contents of an intercepted IP packet. This is the data payload that was contained in the IP packet. It does not include any data from the packet header.

Message Header 2: 1203 contains the next Message Header in the Packet Driver buffer.

Data contents 2: 1204 contains the next IP packet data contents in the Packet Driver buffer.

(up to)

Message Header n: 1205 contains the last Message Header in the Packet Driver buffer.

Data contents n: 1206 contains the last IP packet data contents in the Packet Driver buffer.

The Message Header and data contents sequence can be repeated up to the limits of the available space in the buffer.

FIGURE 9A illustrates the contents of the TPO Packet Driver Routing Header. It consists of the following fields:

Function type: 1210 indicates the type of data in the Packet Driver buffer:

DATA:	indicates normal packet data
NOTIFY:	indicates a notification message used for Packet Driver flow control

Number of messages: 1211 specifies the number of Message Headers contained in the Packet Driver buffer.

Data length: 1212 specifies the total length of all data in the Packet Driver buffer following the Routing Header.

FIGURE 9B illustrates the contents of the TPO Packet Driver Message Header. It consists of the following fields:

Version number: 1220 indicates the version number of the Packet Driver protocol.

Length of this header: 1221 specifies the length of this Message Header.

Message function: 1222 indicates the type of this message:

DATA:	indicates normal packet data
FLOW:	indicates a flow control message
CONNREQ:	indicates a user application connection request
CONNRESP:	indicates a user application connection respons
ERROR:	indicates a Packet Driver error message

Message flag: 1223 specifies flags:

FSTART:	indicates the start of flow control
FSTOP:	indicates the stop of flow control

Protocol type: 1224 specifies the protocol type of the intercepted IP packet:

TCP:	indicates the intercepted packet was the TCP protocol type
UDP:	indicates the intercepted packet was the UDP protocol type
ICMP:	indicates the intercepted packet was the ICMP protocol type

Message sequence number: 1225 specifies a sequence number assigned by the Packet Driver to this Message Header. These sequence numbers are maintained relative to each intercepted IP stream. For example, the Packet Driver buffer could contain the following Message Headers:

IP packet 1 for intercepted quintuple 1* (src:p,dst:p,prot)	Message Sequence 1
IP packet 2 for intercepted quintuple 1* (src:p,dst:p,prot)	Message Sequence 2
IP packet 1 for intercepted quintuple 2* (src:p,dst:p,prot)	Message Sequence 1
IP packet 3 for intercepted quintuple 1* (src:p,dst:p,prot)	Message Sequence 3
IP packet 2 for intercepted quintuple 2* (src:p,dst:p,prot)	Message Sequence 2
IP packet 1 for intercepted quintuple 3* (src:p,dst:p,prot)	Message Sequence 1

Intercepted source IP address: 1226 specifies the source IP address of the intercepted IP packet.

Intercepted destination IP address: 1227 specifies the destination IP address of the intercepted IP packet.

Intercepted source port number: 1228 specifies the source port number of the intercepted IP packet.

Intercepted destination port number: 1229 specifies the destination port number of the intercepted packet.

Length of data: 1230 specifies the total length of data following this header.

Status code: 1231 specifies the completion status of Packet Driver requests.

* Collectively, fields 1226, 1228, 1227, 1229, and 1224 are referred to as a "quintuple".

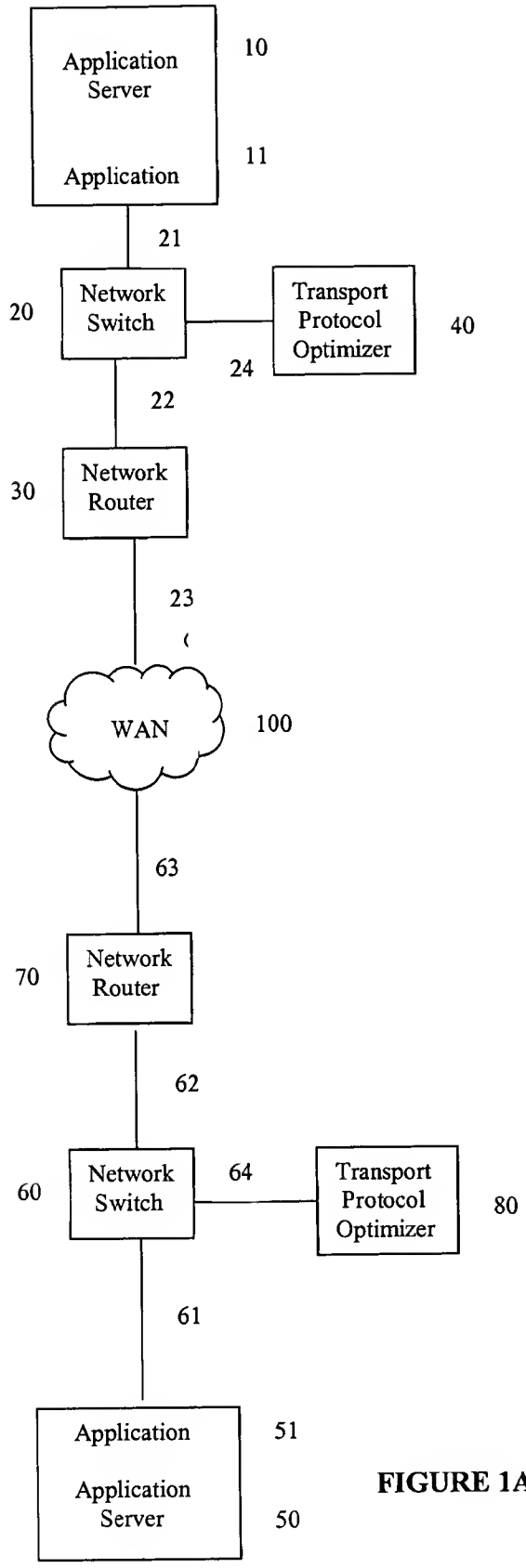


FIGURE 1A. Transport Protocol Optimizer configuration

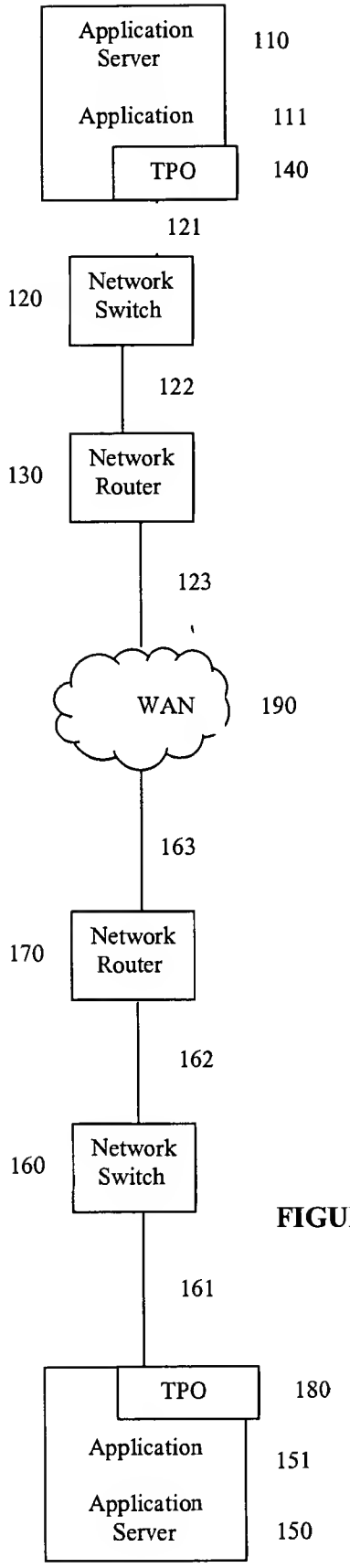


FIGURE 1B. Transport Protocol Optimizer on Application Servers

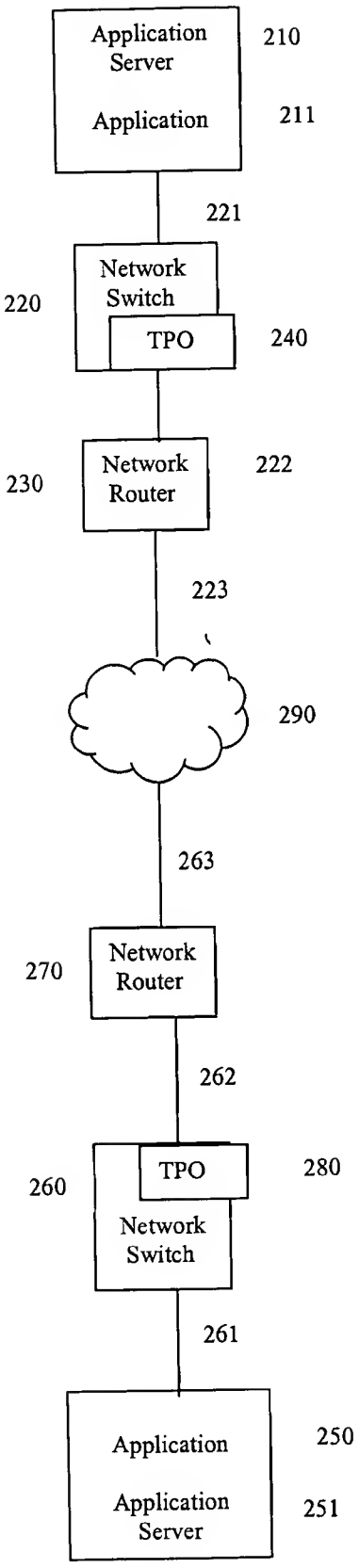


FIGURE 1C. Transport Protocol Optimizer in Switch

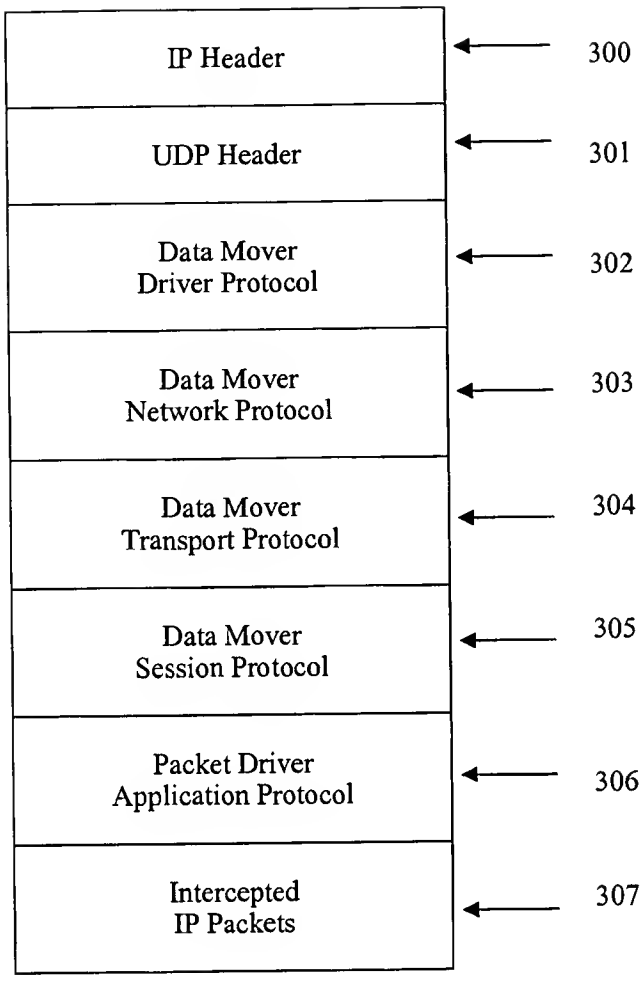


FIGURE 2. Transport Protocol Optimizer Data Unit

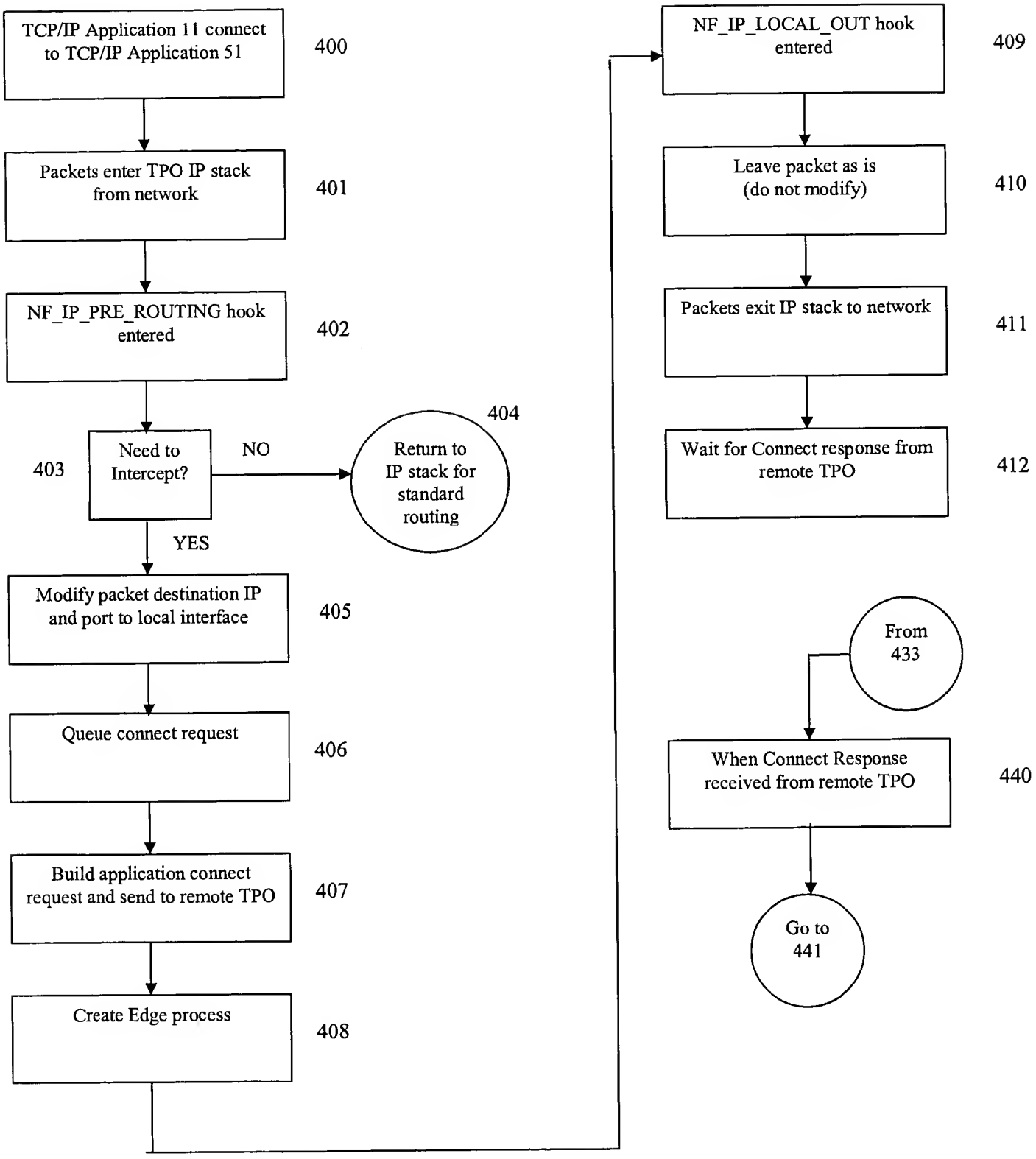


FIGURE 3A. TCP/IP Application Connection Process (1 of 4)

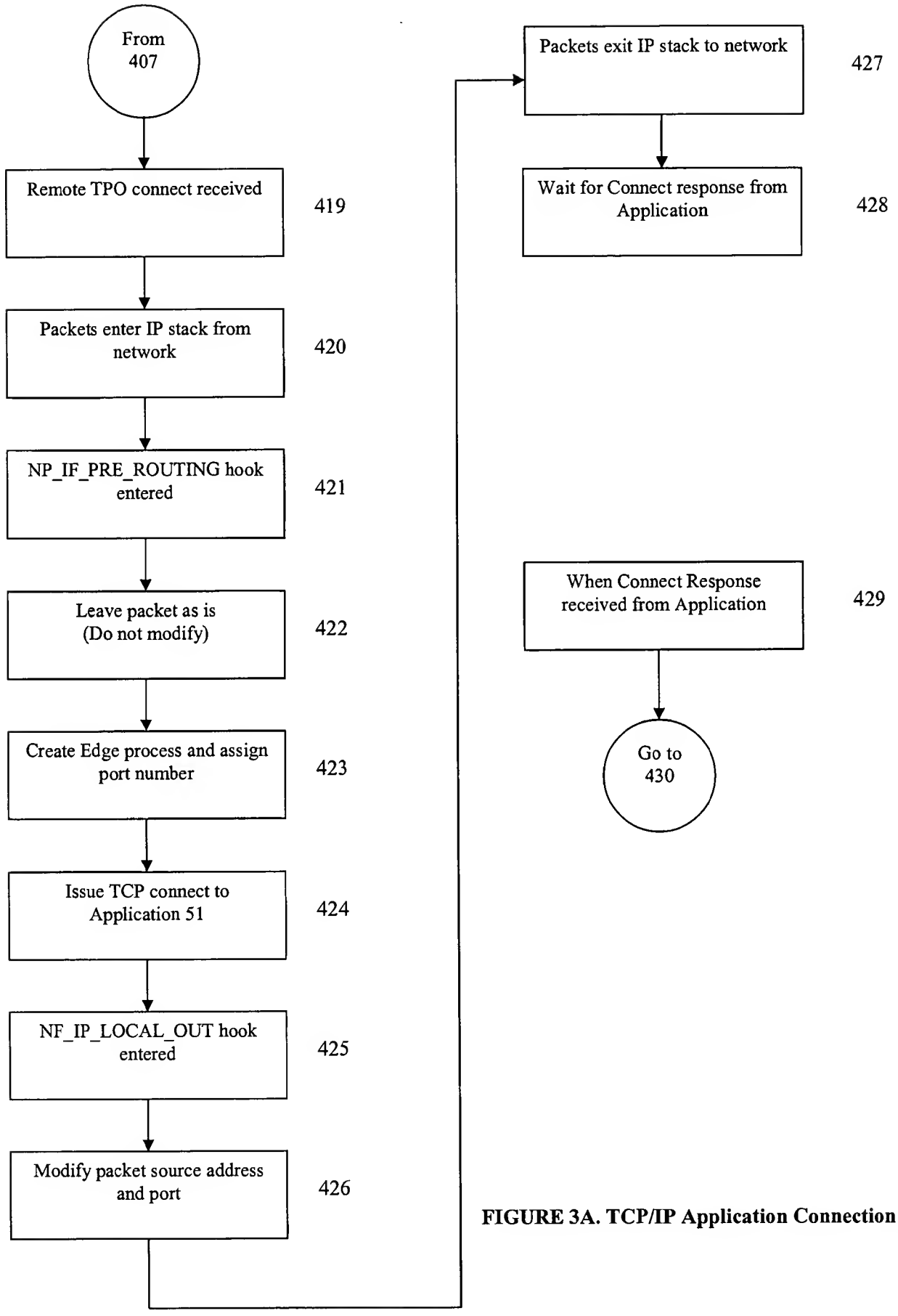


FIGURE 3A. TCP/IP Application Connection Process (2 of 4)

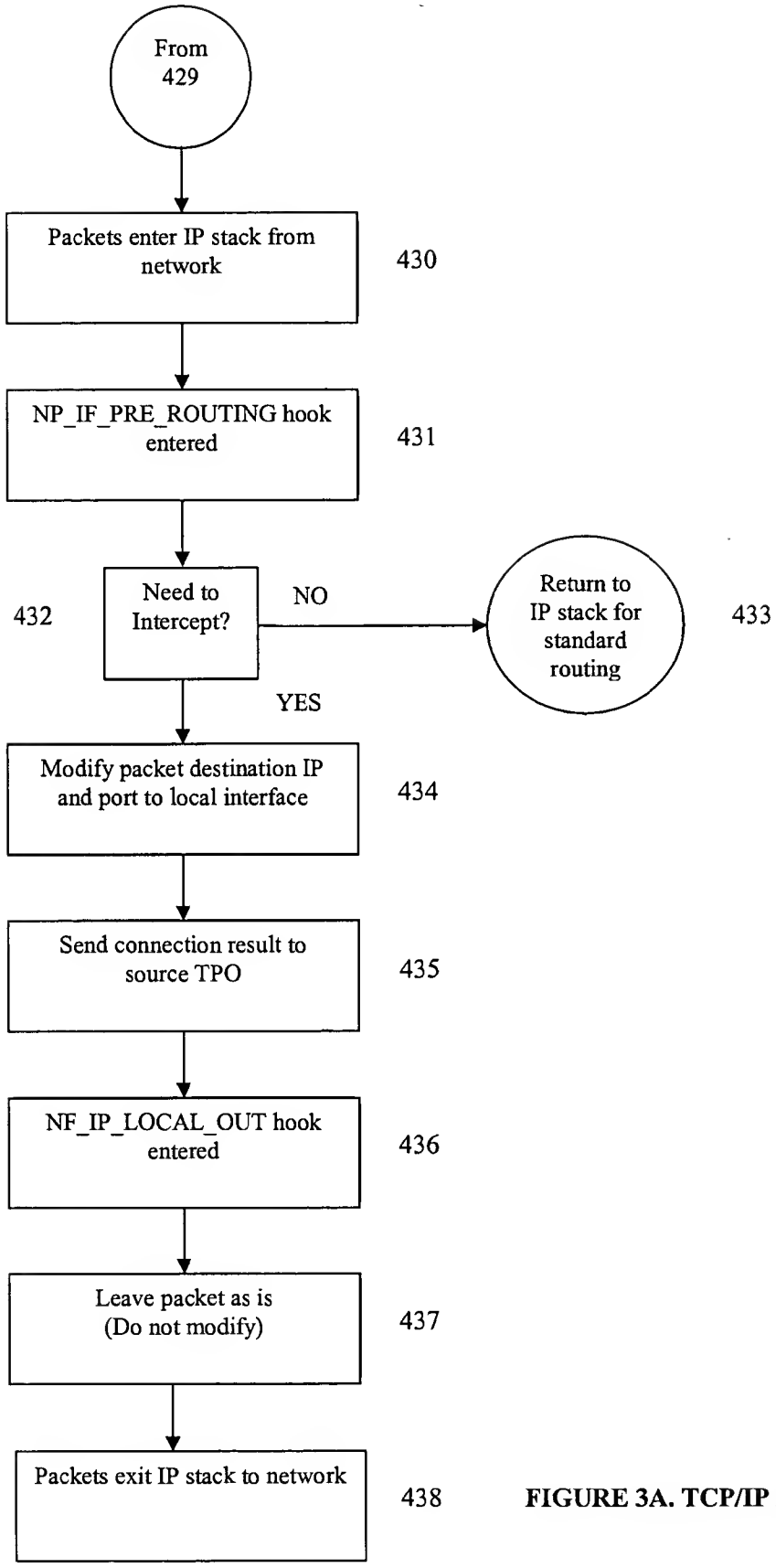


FIGURE 3A. TCP/IP Application Connection Process (3 of 4)

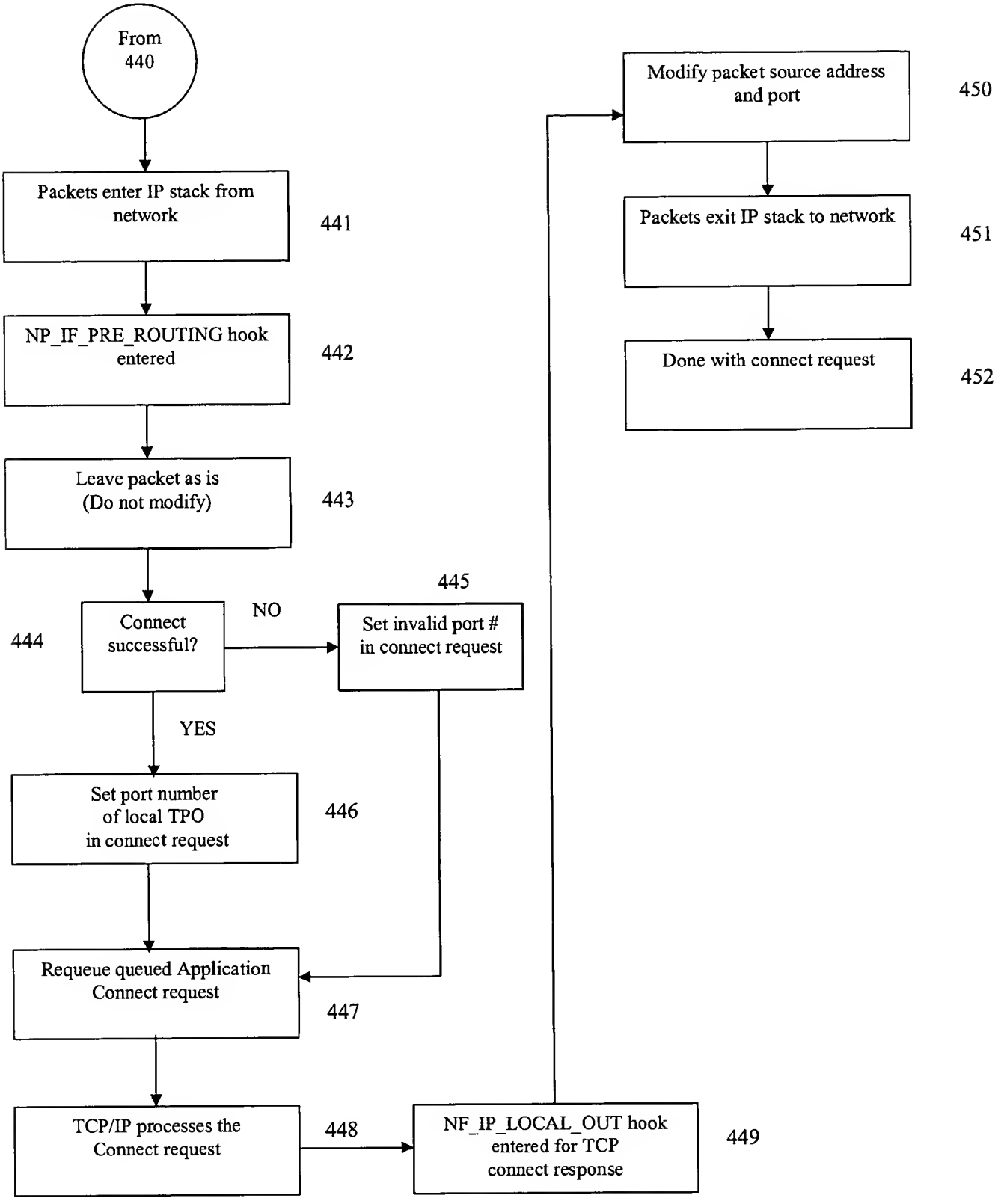


FIGURE 3A. TCP/IP Application Connection Process (4 of 4)

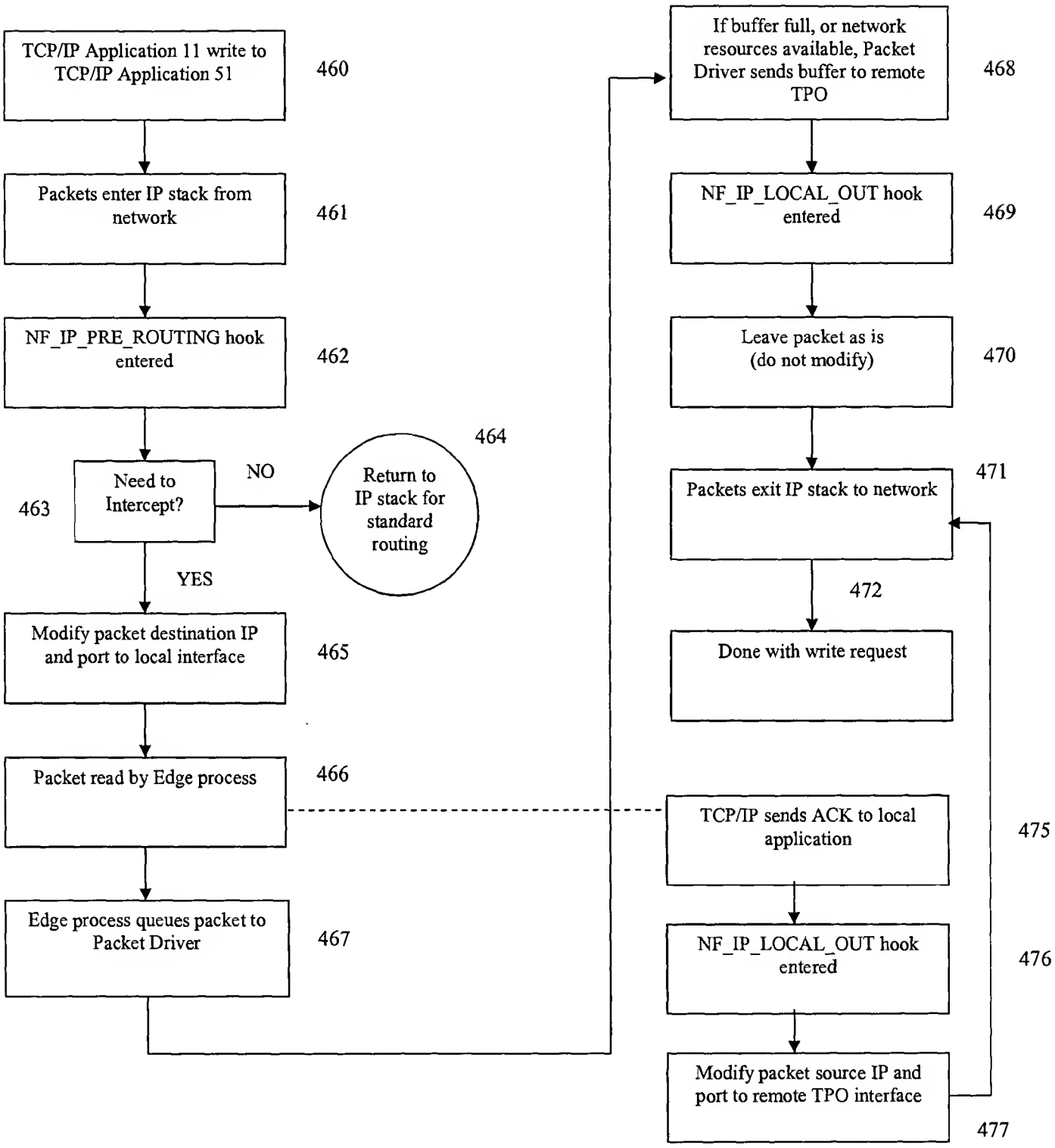


FIGURE 3B. TCP/IP Application Write Process (1 of 2)

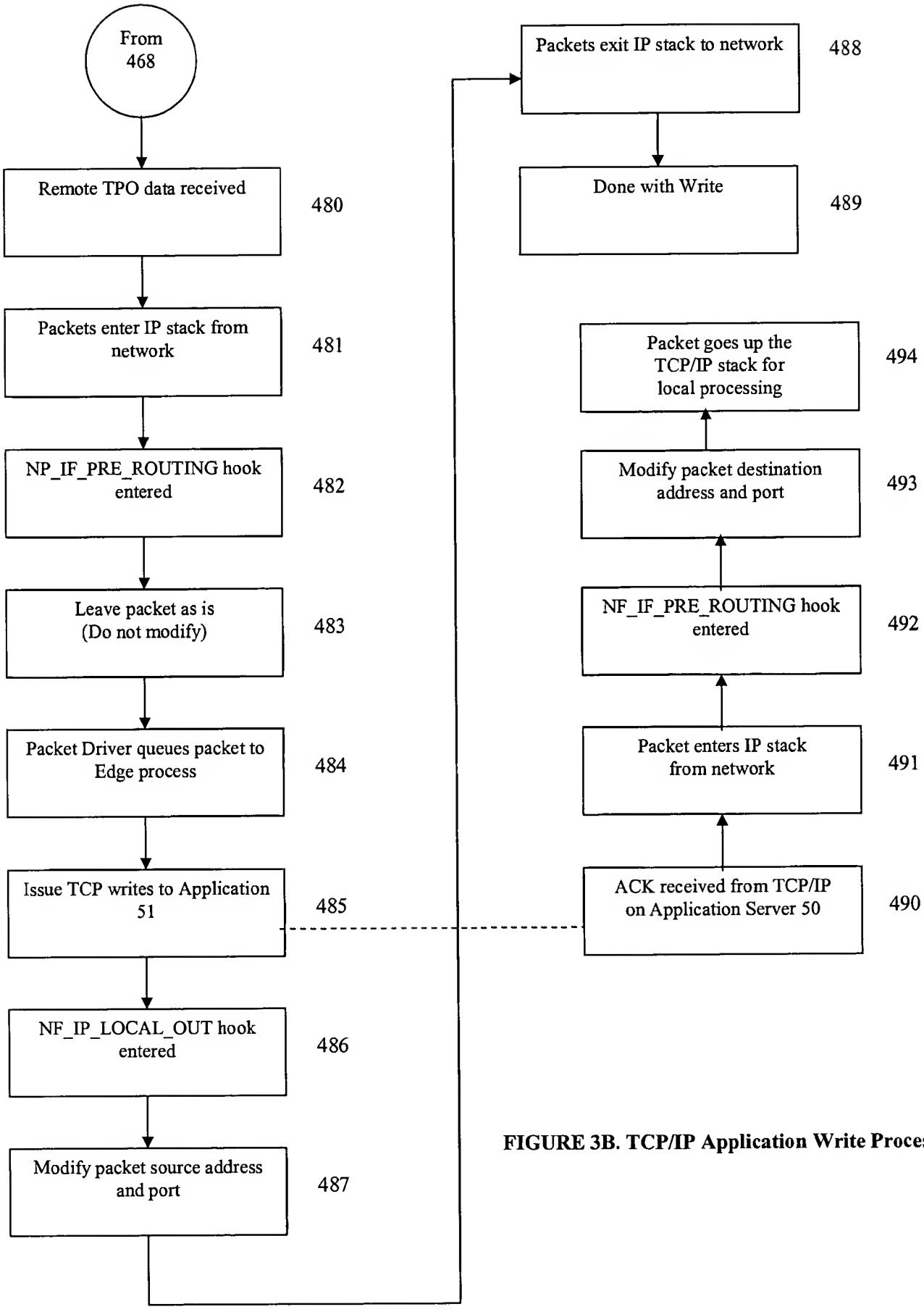


FIGURE 3B. TCP/IP Application Write Process (2 of 2)

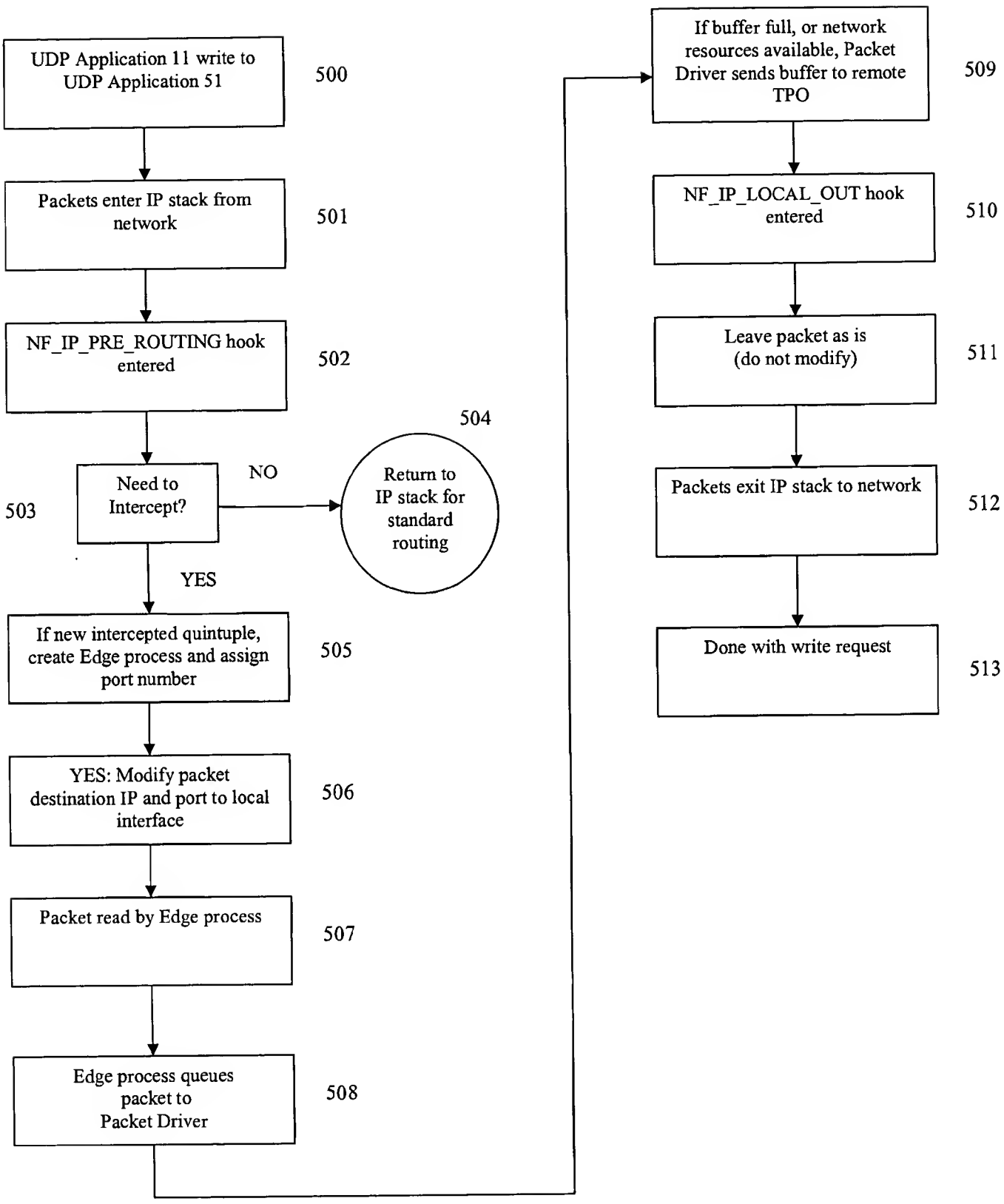


FIGURE 3C. TCP/IP Application UDP Write Process (1 of 2)

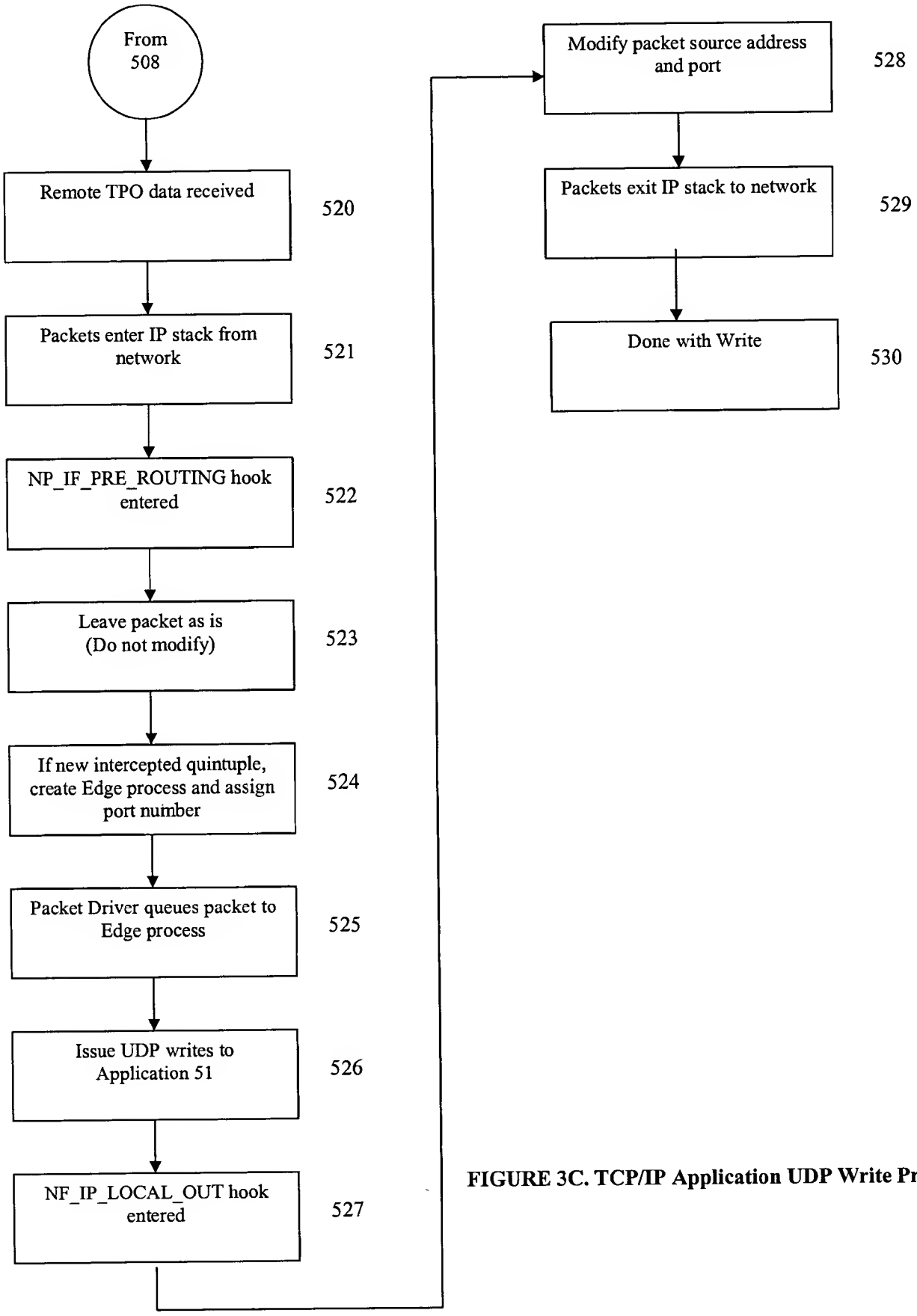


FIGURE 3C. TCP/IP Application UDP Write Process (2 of 2)

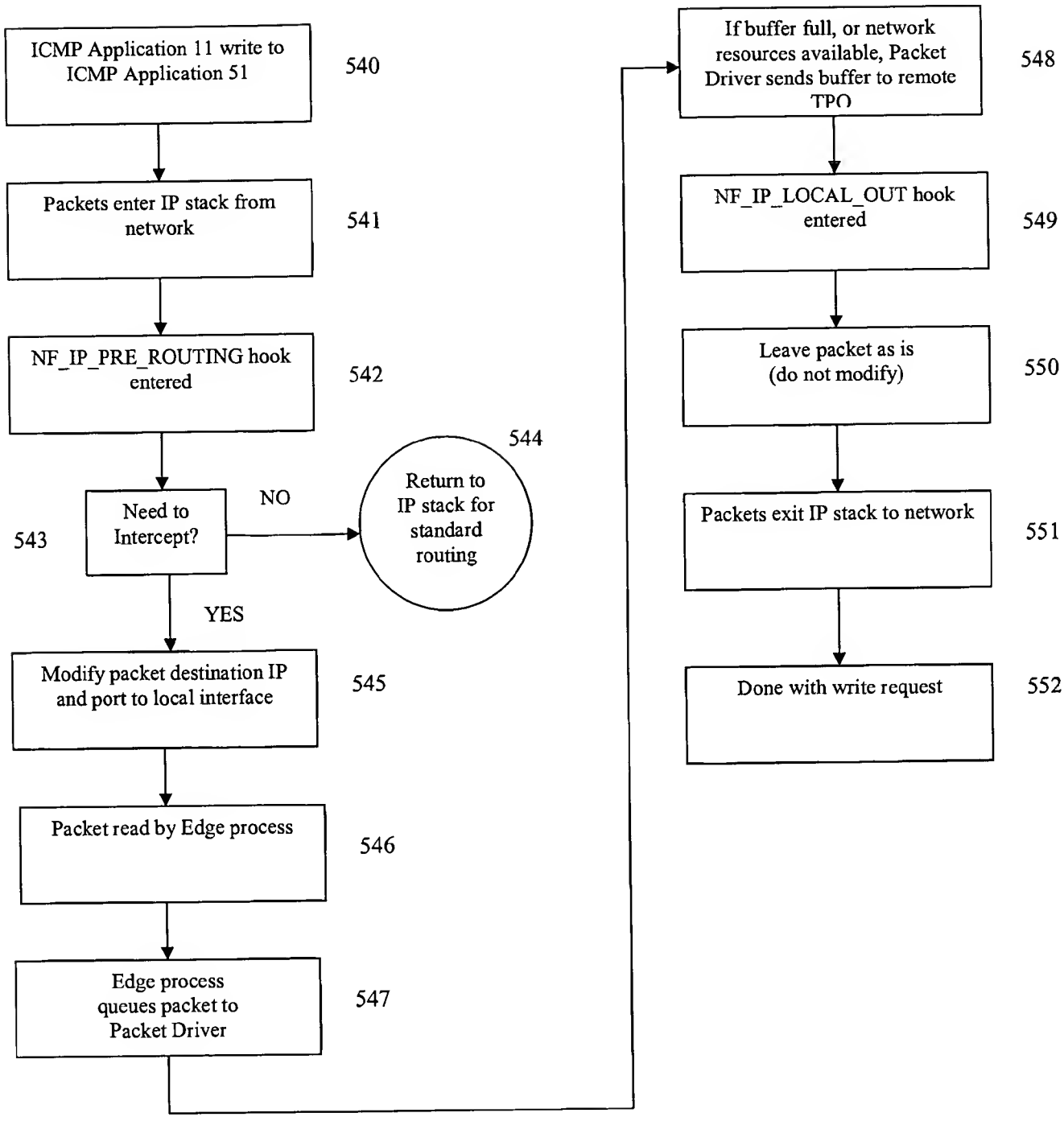


FIGURE 3D. TCP/IP Application ICMP Write Process (1 of 2)

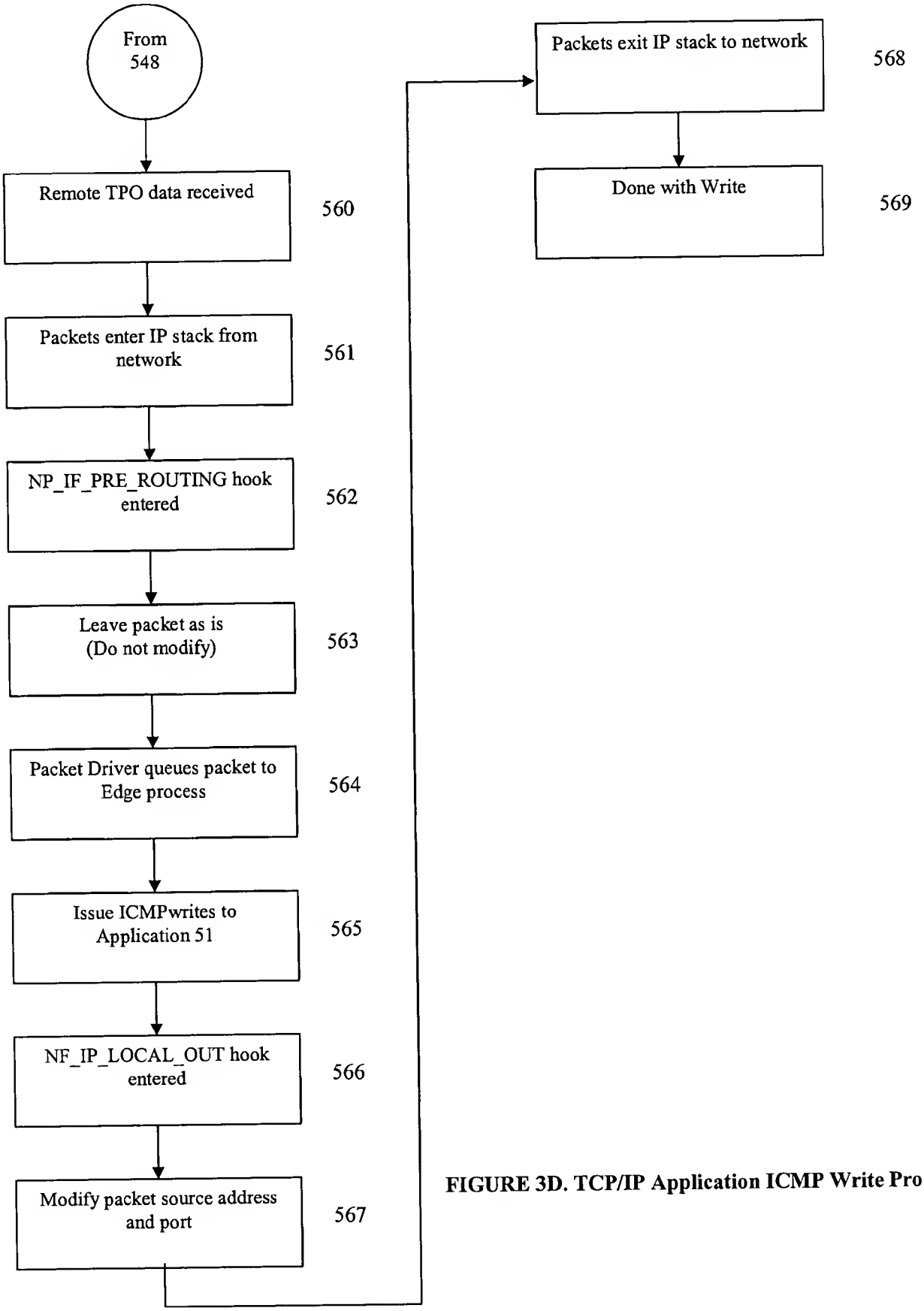


FIGURE 3D. TCP/IP Application ICMP Write Process (2 of 2)

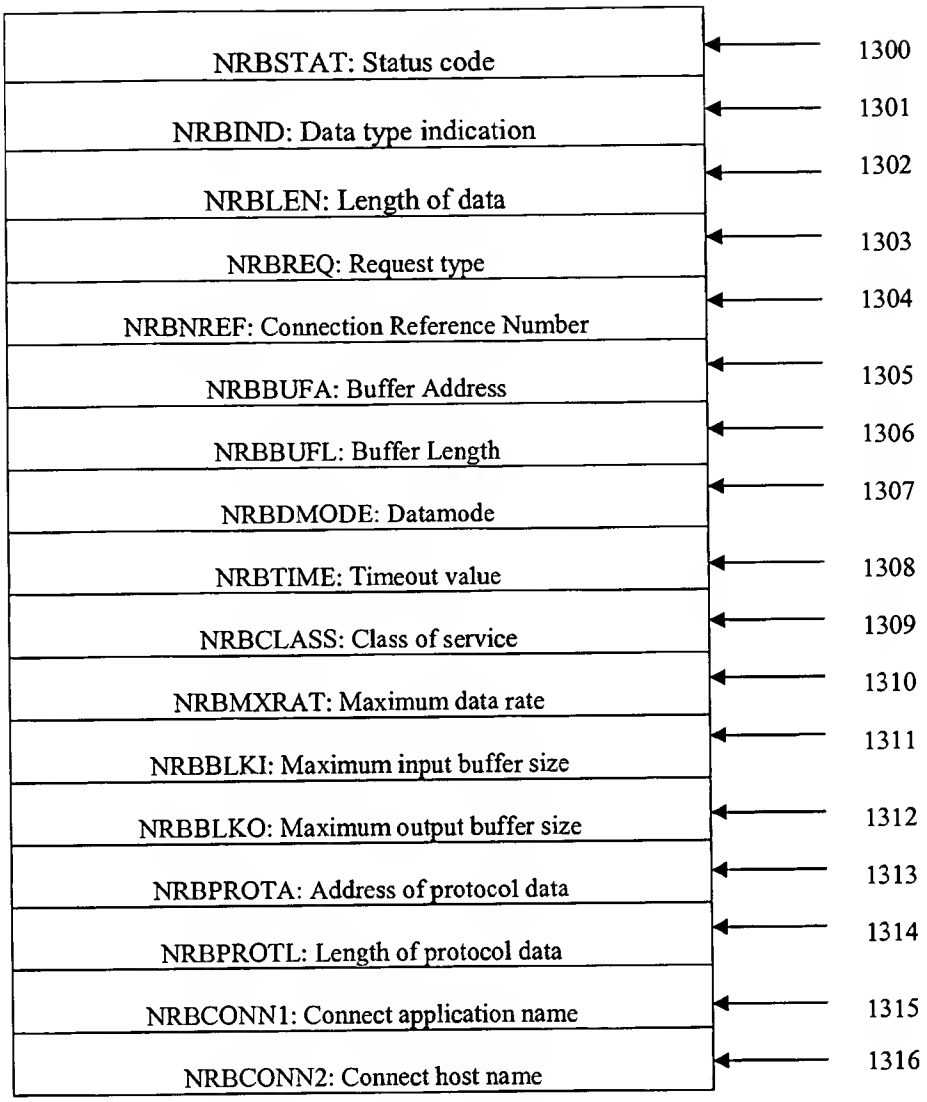


FIGURE 4. Data Mover Network Request Block Structure

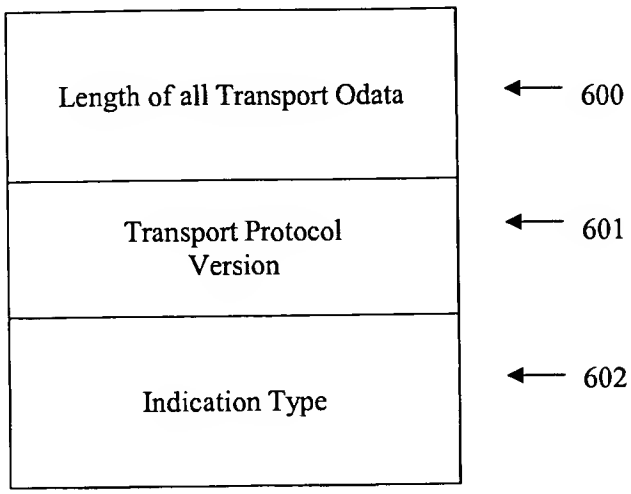


FIGURE 5. Driver Protocol Base Field - ***to be updated***

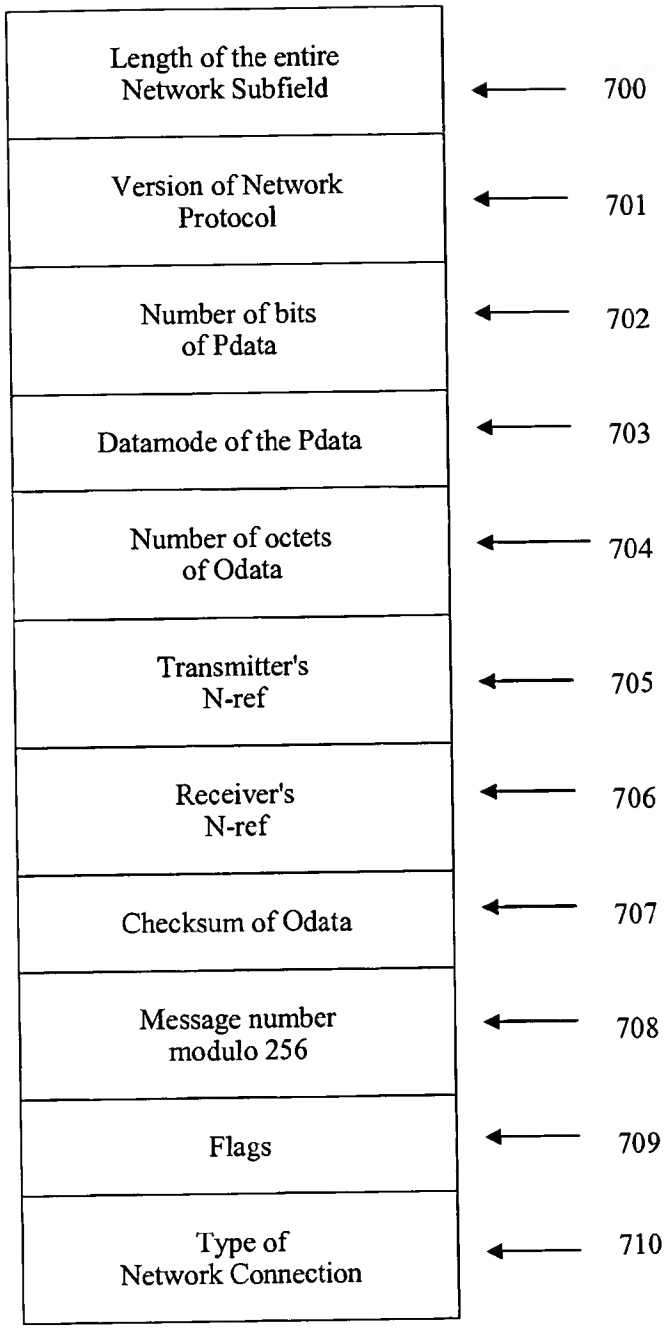


FIGURE 6A. Network Message Header

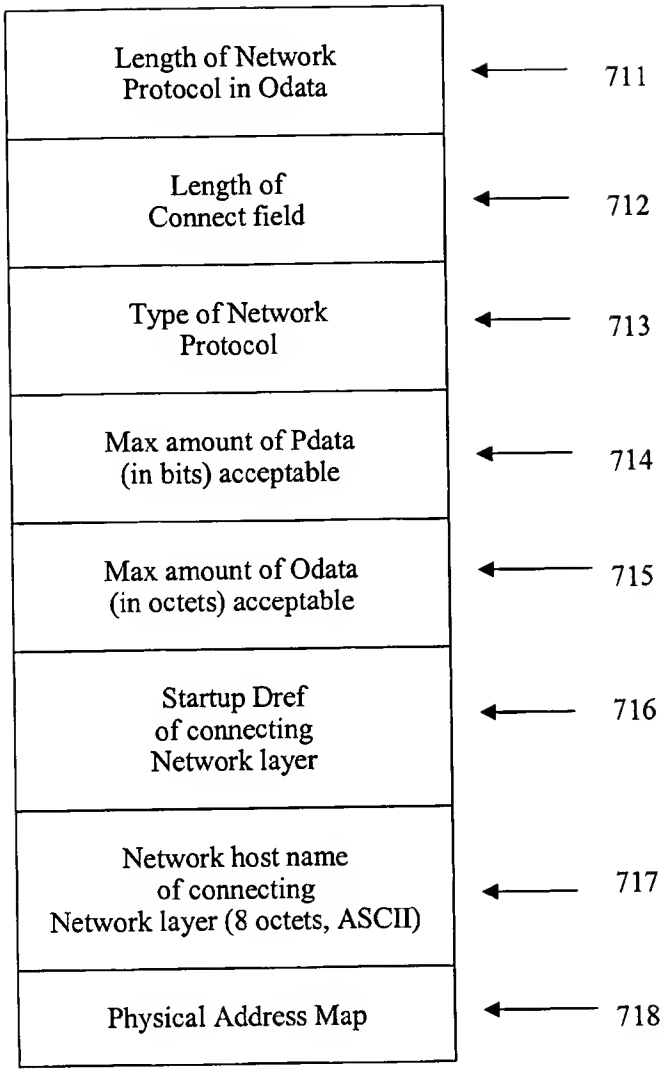


FIGURE 6B. Network Connect / Confirm Protocol Data

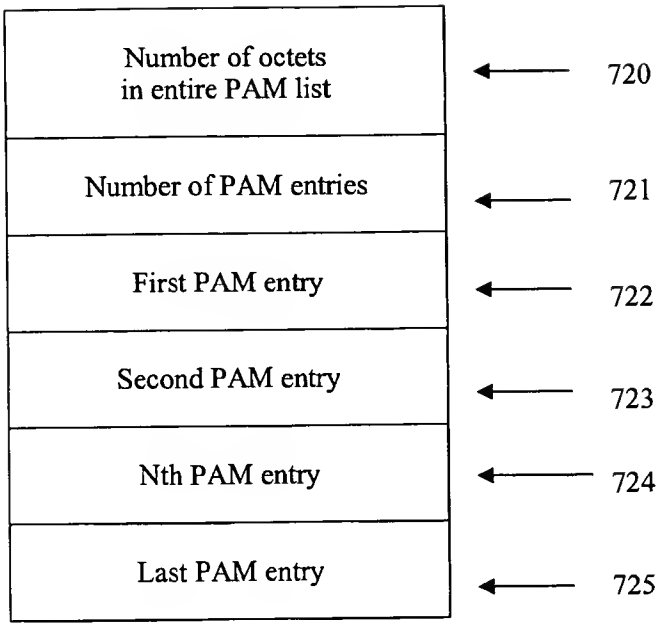


FIGURE 6C. Physical Address Map List

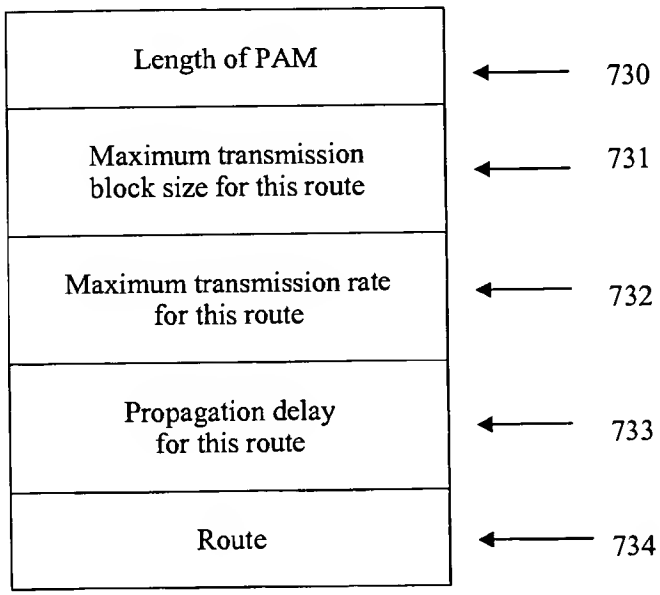


FIGURE 6D. Physical Address Map Entry

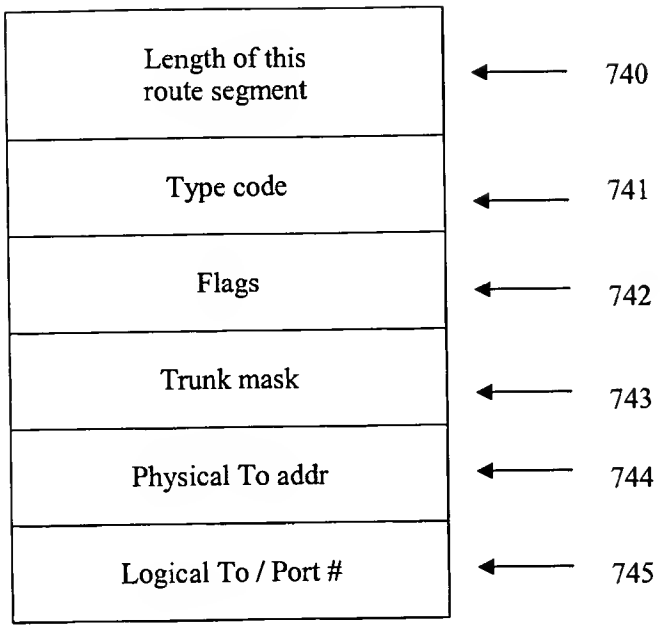


FIGURE 6E. Physical Address Map Route

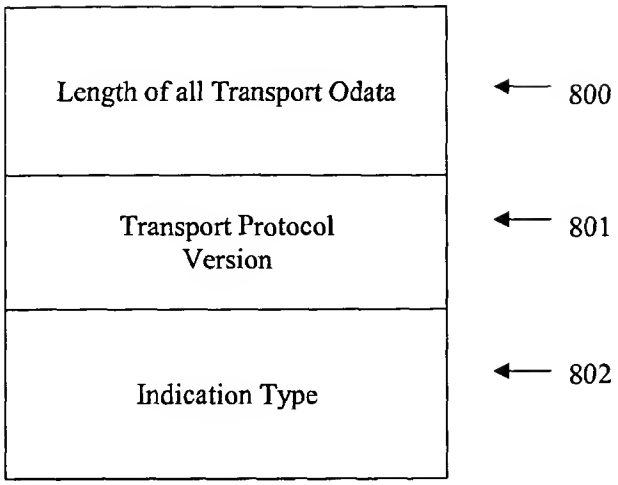


FIGURE 7A. Transport Protocol Base Field

The Transport Protocol Base Field is followed by one or more Transport Protocol Subfields (Figures 7B through 7E).

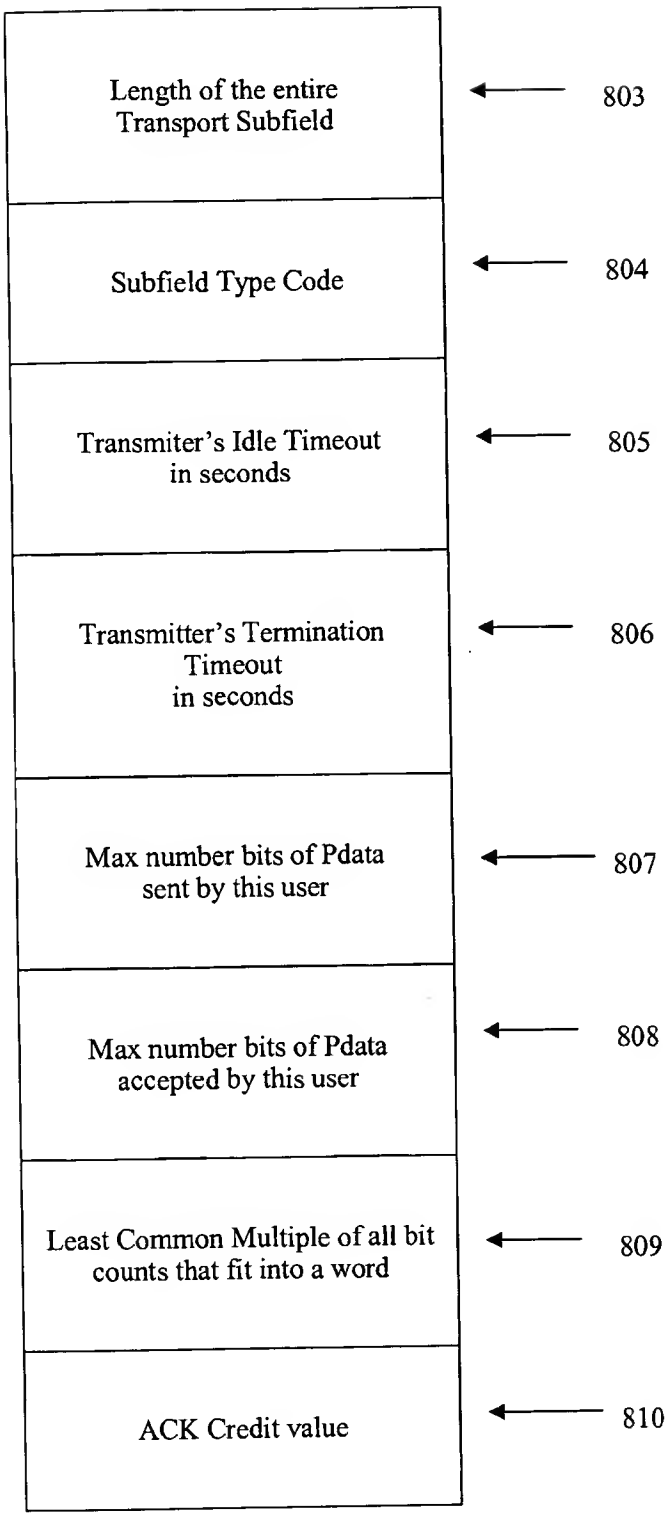


FIGURE 7B. Transport Protocol Connect Subfield

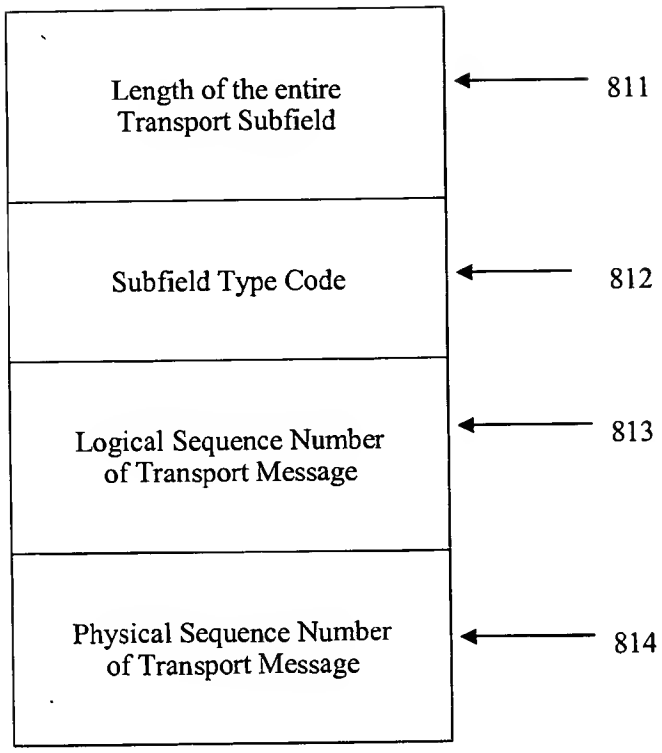


FIGURE 7C. Transport Protocol Data Subfield

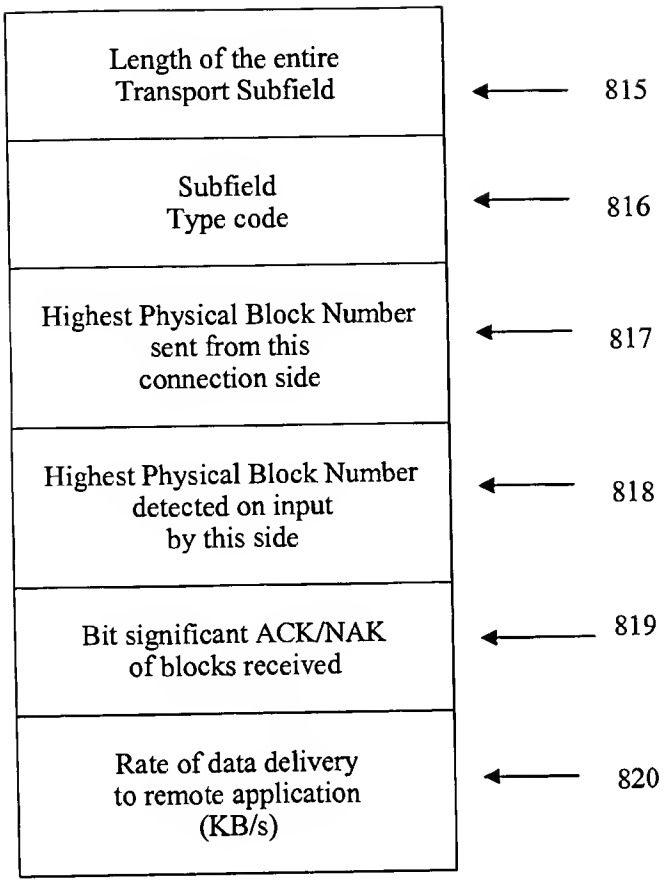


FIGURE 7D. Transport Protocol Acknowledgement Subfield

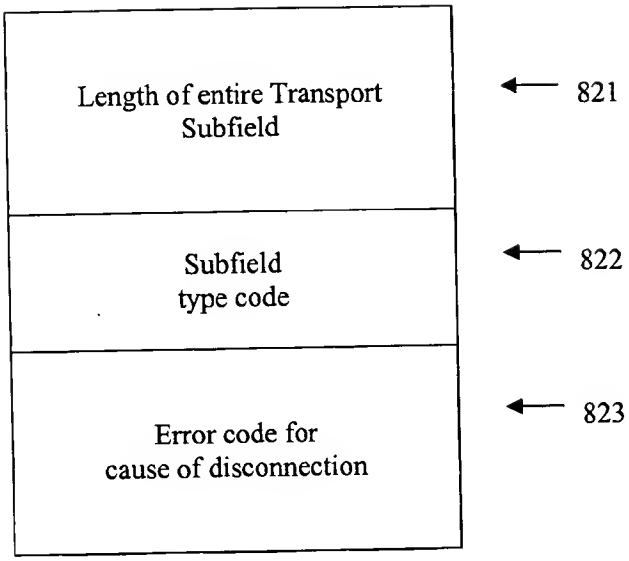


FIGURE 7E. Transport Protocol Disconnect Subfield

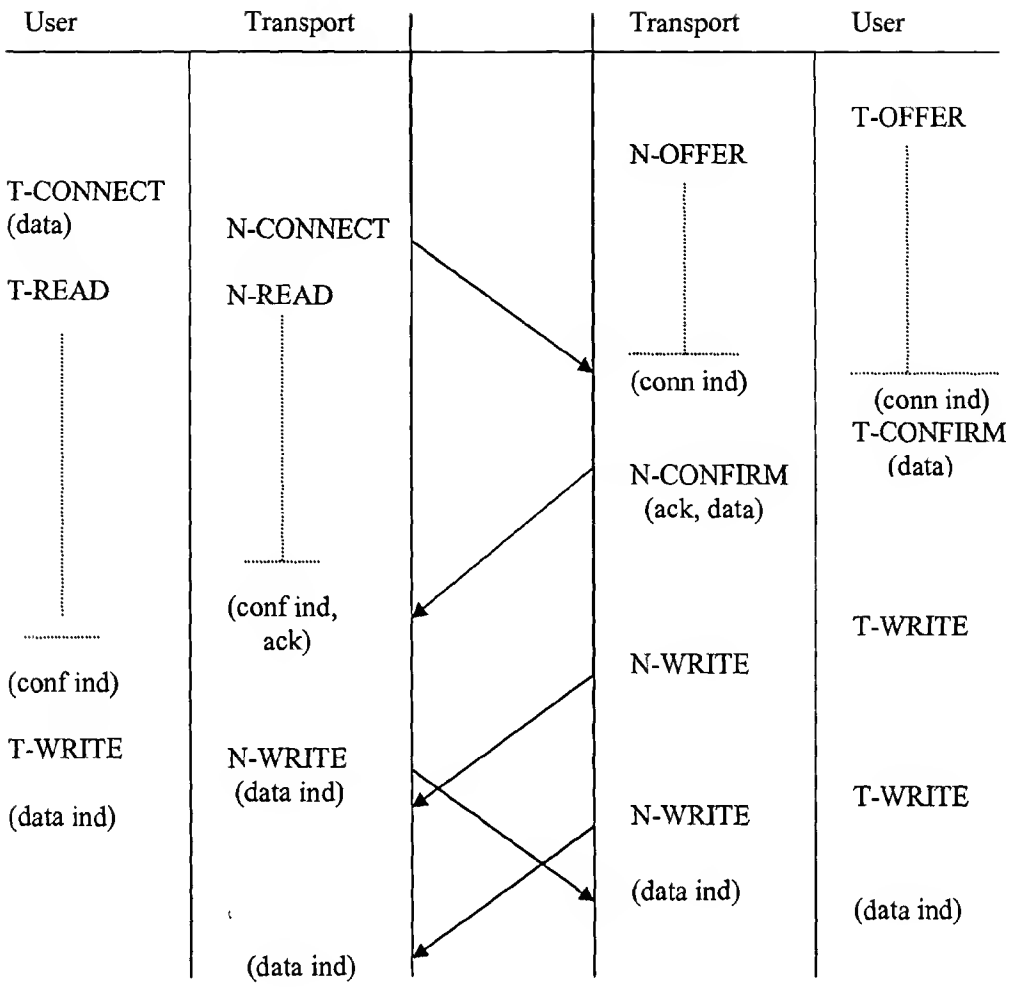


FIGURE 7F. Transport Connection Sequence

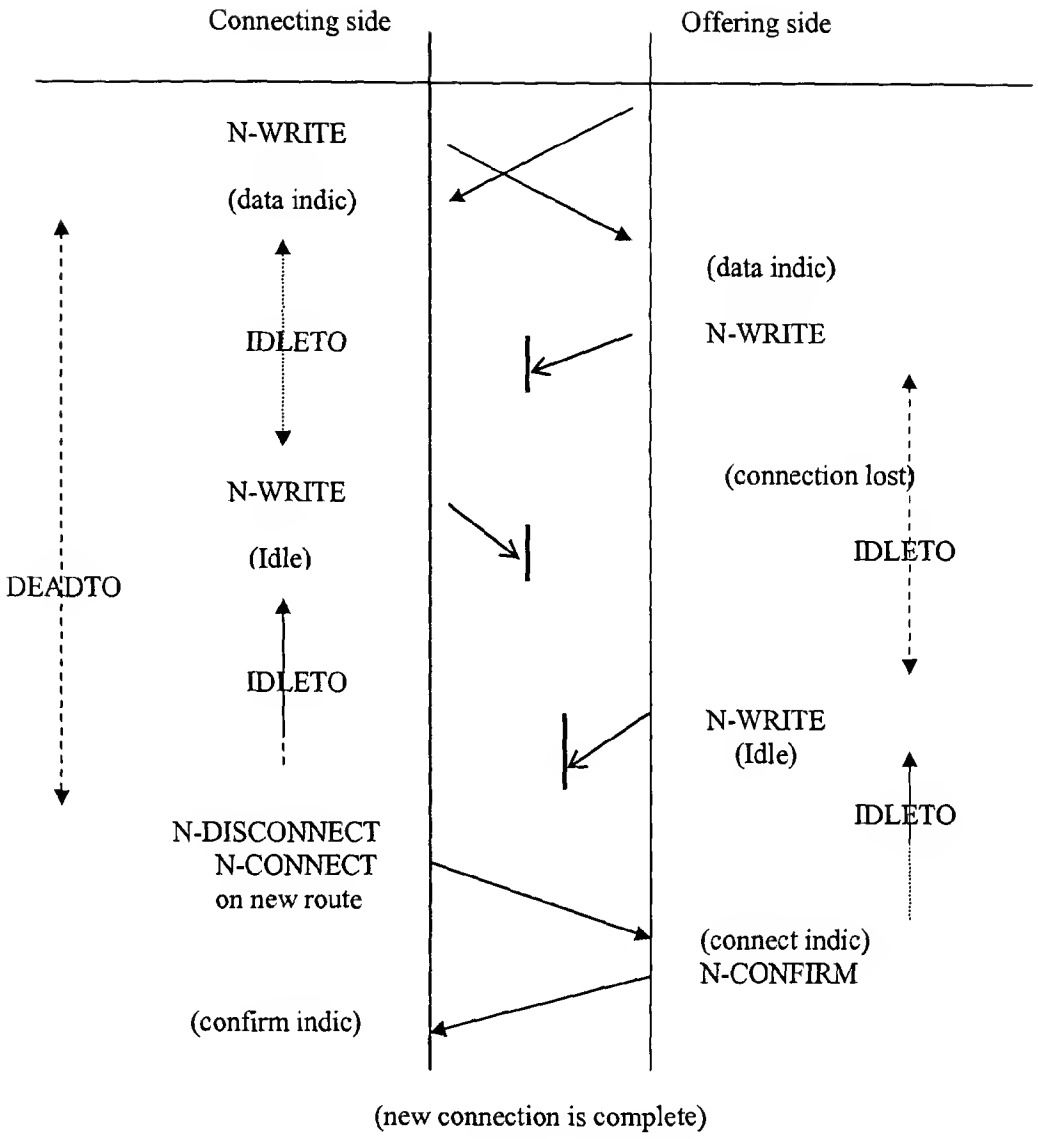


FIGURE 7G. Transport Reconnection Sequence

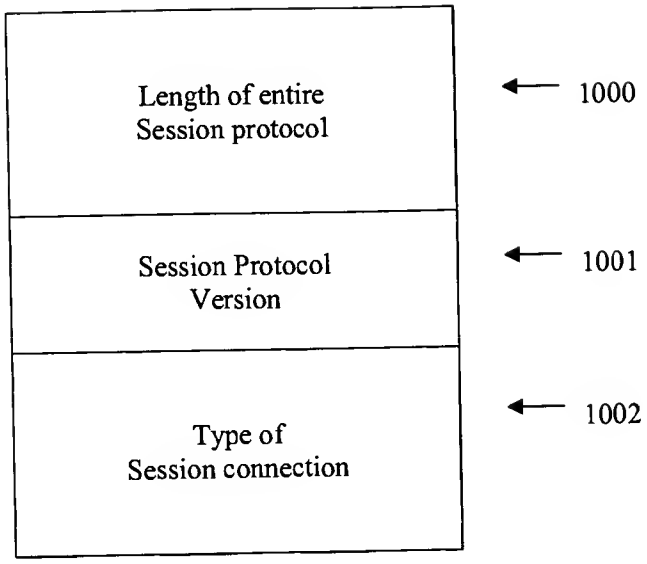


FIGURE 8A. Session Manager Protocol Base Field

The Session Manager Protocol Base Field is followed by either the Session Manager Protocol Connect or Confirm Subfields (Figures 8B through 8C).

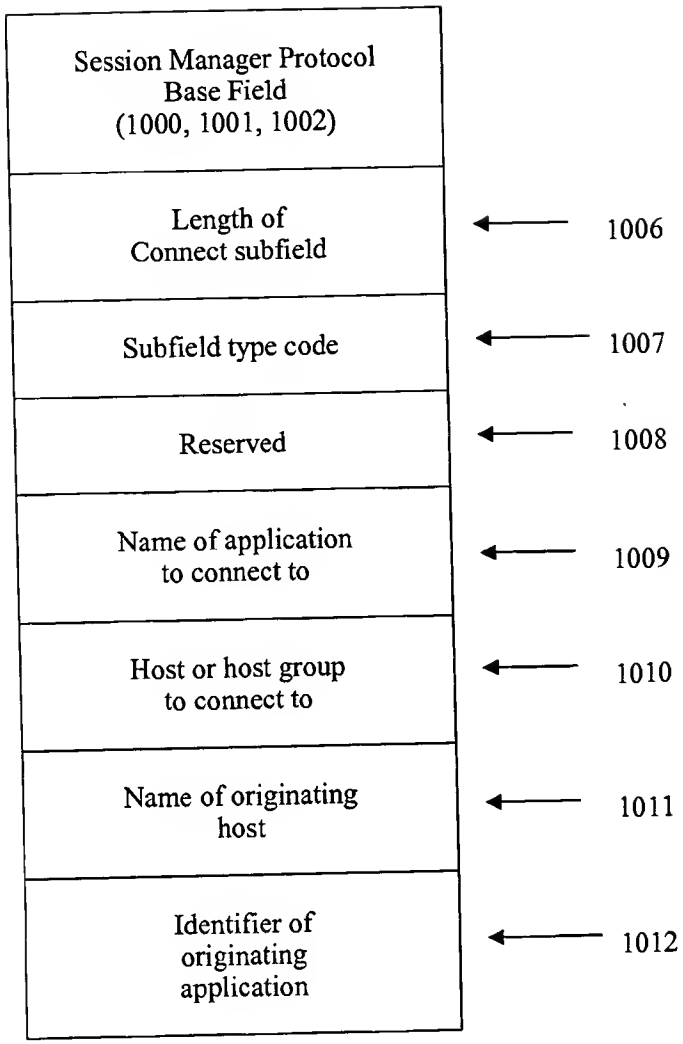


FIGURE 8B. Session Manager Protocol Connect Subfield

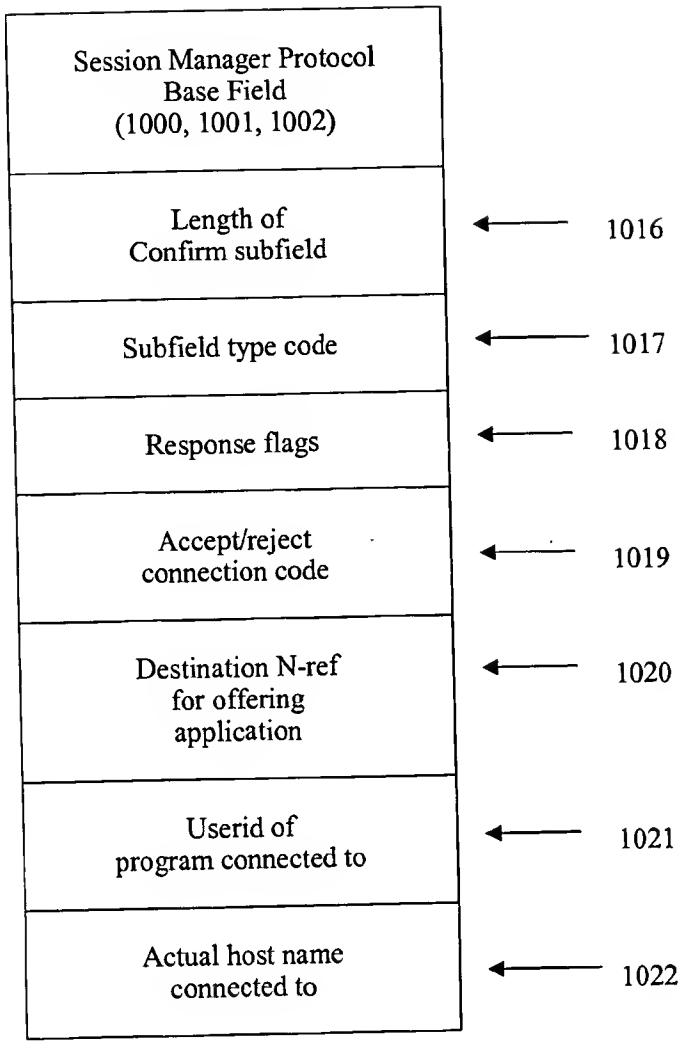


FIGURE 8C. Session Manager Protocol Confirm Subfield

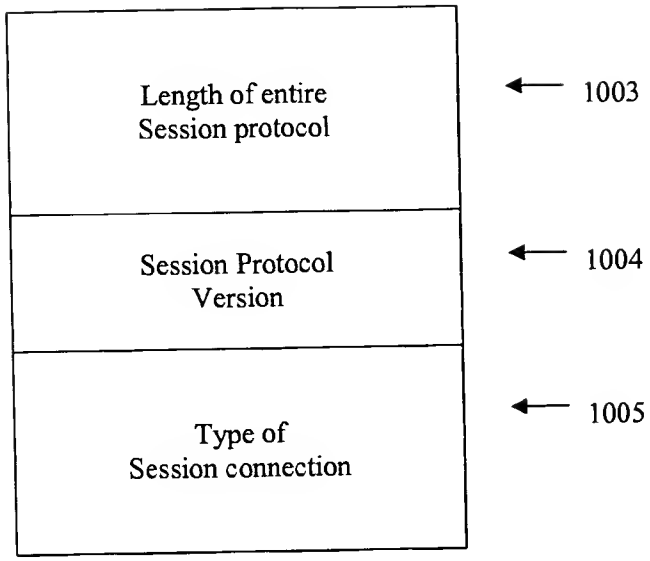


FIGURE 8D. Session Services Protocol Base Field

The Session Services Protocol Base Field is followed by the Session Services Protocol Null Subfield (Figure 8E).

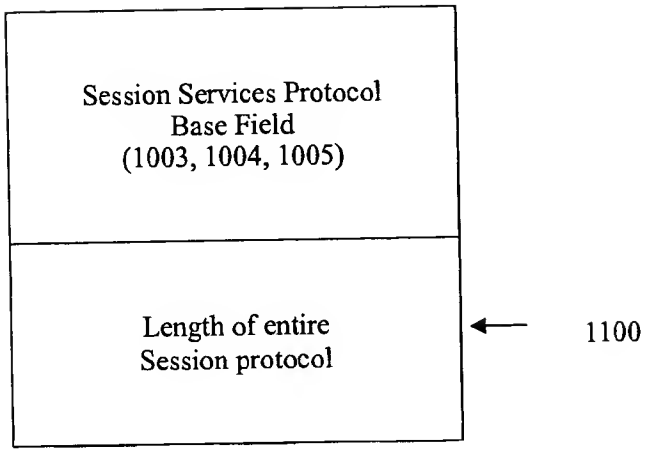


FIGURE 8E. Null Session Protocol Subfield

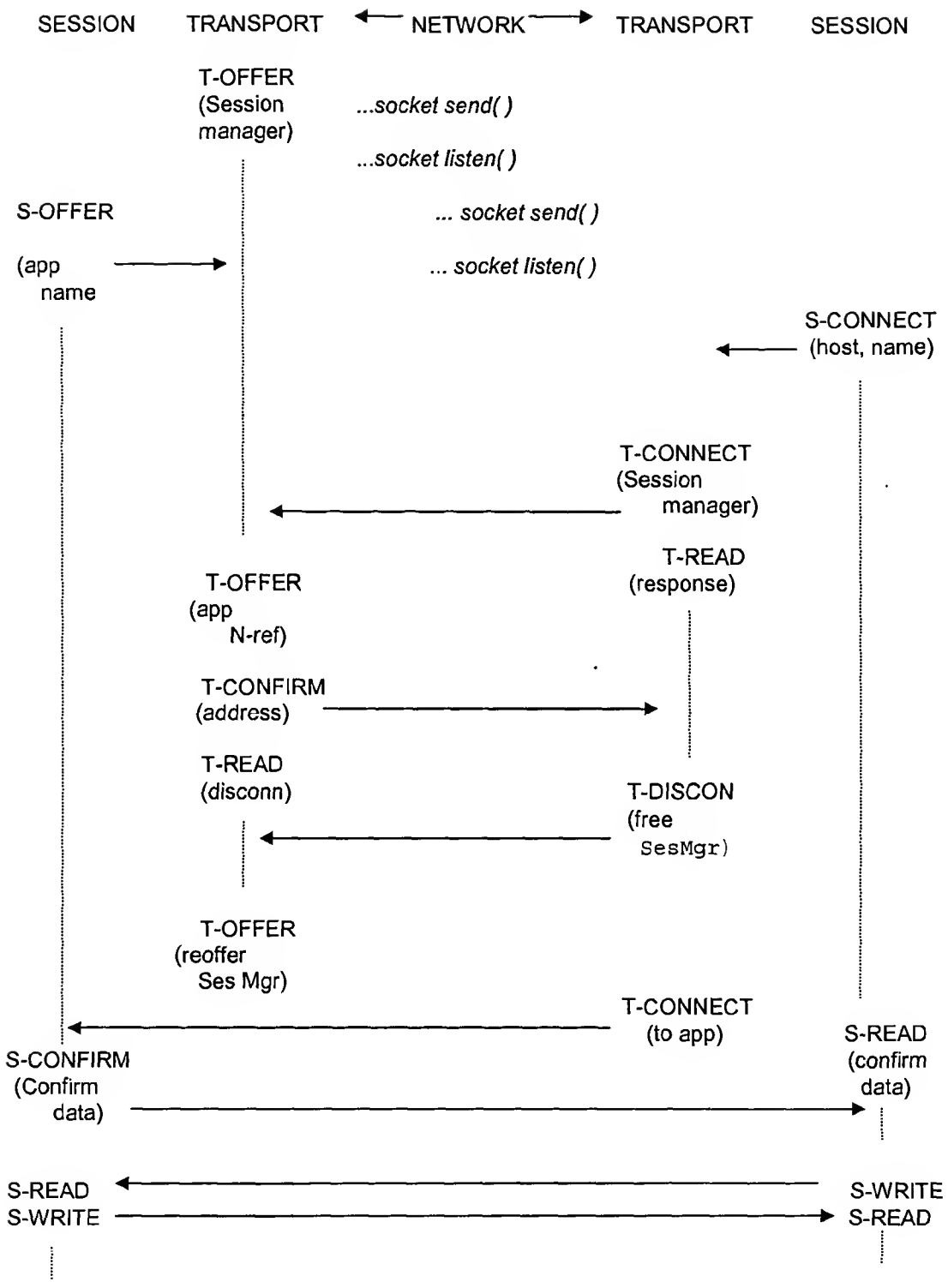


FIGURE 8F. TPO Session Connection Sequence

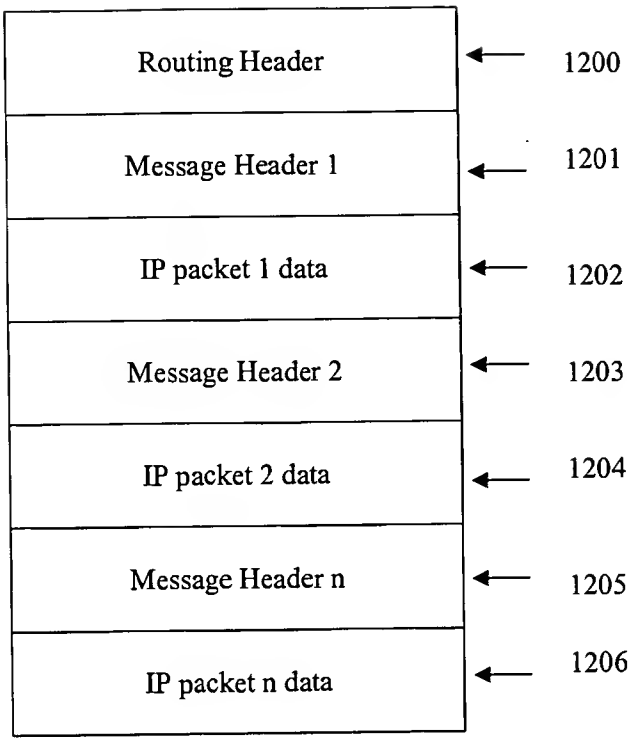


FIGURE 9. Packet Driver Protocol Data

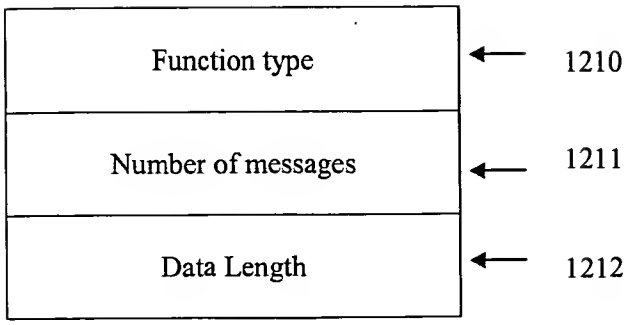


FIGURE 9A. Packet Driver Protocol Data - Routing Header

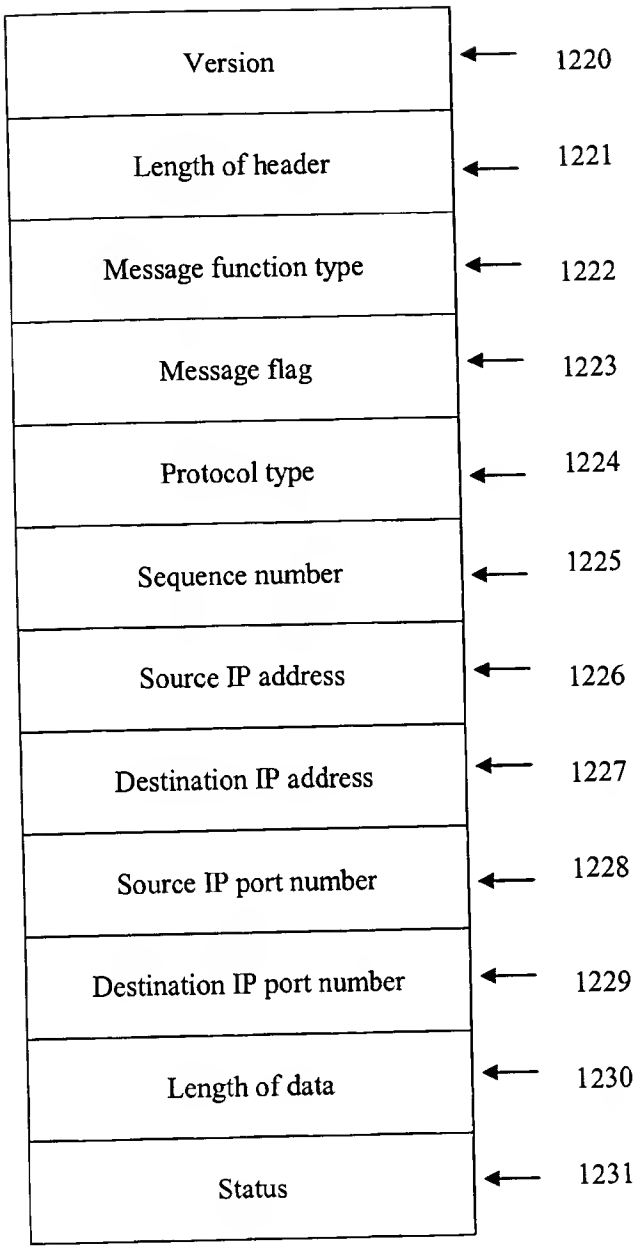


FIGURE 9B. Packet Driver Protocol Data - Message Header